VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Počítačové zpracování fotografie

Computer processing of picture

2009

Lukáš Krahulec

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. květen 2009 .....

Rád bych poděkoval zejména doc. Ing. Lačezaru Ličevovi, CSc. za trpělivé a ochotné vedení této práce a za jeho cenné rady a připomínky při vývoji aplikace. Dále bych rád poděkoval Bc. Janu Královi za spolupráci při analýze a vývoji jádra programu.

Abstrakt

Cílem této diplomové práce je seznámit se s fotogrammetrickým systémem Fotom 2007, analyzovat a zhodnotit jeho možnosti. Dále na základě požadavků na moderní systém zvolit vhodné programové prostředky a vytvořit nové jádro pro systém Fotom 2009. S jádrem systému vznikne robustní API podporující snadnou rozšiřitelnost systému a jednoduché sdílení informací mezi moduly. Dále na vytvořeném API vzniknou moduly pro definici zájmových objektů na snímku, jejich měření. Pro snazší detekci objektů vzniknou moduly pro skicování, tedy úpravy snímků pomocí filtrů a kreslicích nástrojů. Tato práce si bere za cíl vytvořit novou generaci fotogrammetrického systému Fotom s jednoduchým uživatelským rozhraním a snadnou rozšiřitelností pomocí modulů.

Klíčová slova: fotogrammetrie, Fotom 2008, Netbeans platform, architektura, Java, zpracování snímku, prahování, filtr, zájmový objekt, xml

Abstract

The goal of this dissertation is get acquainted with photogrammetric system Fotom 2007, analyse and evaluace it's potentialities. Furthermore choose acceptable programmatic resources and create the new core for the system Fotom 2009 on the strength of demands to the modern system. With the system core will be created huge API supporting easy expansibility of the system and also easy sharing of the informations among the modules. In created API will ensue modules to define objects of interest in the image and it's metering. For easier detection of the objects will be created modules for sketching, meant edits of the images via filters and drawing tools.This dessertation want to create new generation of photogrammetric system Fotom with user friendly interface and easy expansibility by modules.

Keywords: photogrammetry, Fotom 2008, Netbeans platform, architecture, Java, image processing, thresholding, filter, object, xml

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
AWT	–	Abstract Window Toolkit
CCD	–	Charge-Coupled Device
CT	–	Computed Tomography
GUI	–	Graphical User Interface
IDE	–	Integrated Development Environment
JAI	–	Java Advanced Imaging
JAR	–	Java Archive
JEE	–	Java Enterprise Edition
JVM	–	Java Virtual Machine
MRI	–	Magnetic resonance imaging
px	–	Pixel, obrazový bod
RGB	–	Barevný model red-green-blue
RCP	–	Rich Client Platform
XML	–	Extensible Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Ve své diplomové práci se zabývám digitálnímu zpracování obrazu za účelem měření. Této problematice se zabývá věda zvaná fotogrammetrie, kterou představím kapitole 2.1. Měření na snímcích se velmi často využívá v hornictví k proměřování důlních šachet a sledování změn jejich profilů v čase. V kapitole 2.2 se budu zabývat aplikací Fotom 2008, nejnovější verze fotogrammetrického systému, který je vyvíjen a rozšiřován na fakultě informatiky FEI VŠB TU Ostrava. Vyvinul se z jednoúčelového programu, který byl nejdříve zaměřen k analýze především snímků šachet k systému, který je vhodný i k měření snímků v medicíně, kde se také v hojné míře využívá fotogrammetrii jako neinvazivní způsob vyšetření.

Stávající verze systému již nevyhovuje požadavkům na moderní systém, protože neumožňuje snadnou rozšiřitelnost, aktualizace a distribuci nových komponent. Aplikace je psána v programovacím jazyce C++ a jádro systému není popsáno programátorskou příručkou, proto dodatečné přidávání funkcí je obtížné. Pro komunikaci mezi moduly chybí společné API, a proto jednotlivé moduly systému jsou samostatné aplikace pracující nad jedním datovým souborem. Aby bylo možné pokračovat v dalším vývoji nových modulů pro zpracování a vizualizaci snímků, je nutné vytvořit nové jádro systému, které bude především vhodně navrženou architekturou podporovat moularitu systému.

Pro tvorbu nového jádra analyzuji možnosti moderních prostředků a zaměřím se na aktivně vyvíjenou platformu pro vývoj aplikací NetBeans Platform, která je vyvíjená spolu s původně českým projektem NetBeans a je převážně podporován firmou Sun Microsystems. V teoretické části (viz. 3.1) popíšu možnosti této platformy, která slouží jako základní stavební kámen při vývoji moderních modulárních systémů. Jelikož fotogrammetrie je v prvé řadě zpracování snímku, budu se zabývat projektem Java Advanced Imaging (JAI), který poskytuje silné nástroje ke zpracování obrazů pro programy psané v programovacím jazyce Java.

Analýzou stávající aplikace a podle nových nároků na moderní systém a požadavků na intuitivní uživatelské prostředí popíši požadavky na program Fotom 2009, který se stane zcela novou verzí stávajícího systému. Bude postaven na platformě NetBeans a pro přístup ke snímku a k jeho zpracování bude využívat JAI. Popsané požadavky analyzuji s ohledem na univerzálnost a snadnou rozšiřitelnost a navrhnu nové jádro aplikace. Společně s jádrem systému vznikne veřejné API (viz. 4.1), které bude popisovat jednotný přístup ke snímku, komunikaci rozšiřujících modulů a definuje rozhraní pro případné nástroje a filtry. Implementací jádra vznikne základ systému umožňující základní práci se snímkem.

Po implementaci a otestování jádra systému se zaměřím na popis požadavků chování modulů pro definici zájmových bodů a objektů na snímku (viz. 4.2). Podle popsaných požadavků analyzuji na základě vytvořeného API sadu nástrojů, které budou sloužit k manuálnímu a částečně automatickému definování zájmových objektů na snímku. Nástroje budou zaměřeny na zpracování ulrazvukových a důlních snímků tak, aby svými možnostmi pokryly a rozšířily funkce prvního modulu systému Fotom2008. Pro potřeby proměřování

vzdáleností mezi definovanými zájmovými objekty navrhnu a implementuji modul pro počítání a vizualizaci vzdálenosti (viz. 4.2.3).

Pro odstranění vad na snímku analyzuji sadu nástrojů pro editaci a filtraci obrazu, které implementuji v samostatném modulu (viz. 4.3). Tento modul bude převážně sloužit k rozlišení mezi popředím, tedy hledanými objekty, a pozadím a také k filtraci snímků zatížených šumem. Při snímání tkání ultrazvukem často dochází k zachycení nekompletní hranice sledovaného objektu a snímky jsou velmi zatíženy rušivým šumem, zvláště při použití ultrazvuku nízké intenzity při vyšetření mozku. Takové snímky je třeba nejprve upravit a připravit k měření eliminací šumu a vhodnými úpravami, jako je prahování, rozlišit mezi popředím a pozadím. Filtry dále doplním nástroji, kterými bude uživatel schopen doplnit chybějící hranice sledovaných objektů.

V závěru systém se všemi instalovanými moduly ověřím na skutečných snímcích z hornictví a lékařství. Po ověření funkčnosti systému na více platfomách vytvořím uživatelskou a programátorskou dokumentaci určenou pro budoucí vývojáře nových modulů, kterou spolu se zdrojovými kódy a samotnou spustitelnou aplikací umístím na internetové stránky.

2 Fotogrammetrie

2.1 O fotogrammetrii

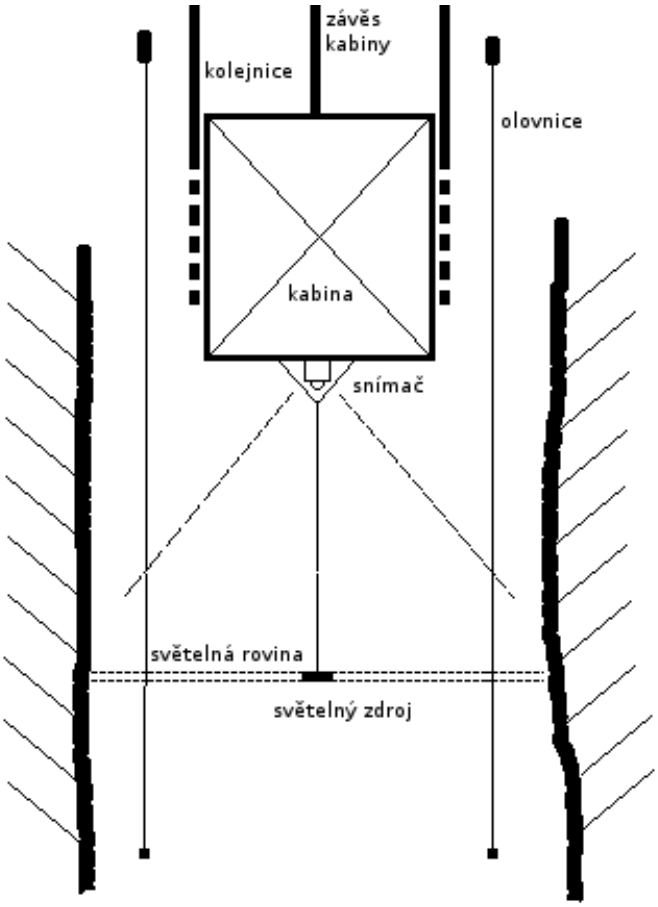
Fotogrammetrie je vědní obor zabývající se zpracováním informací a měřením na fotografických snímcích [5]. Pomocí speciálních přístrojů a systémů lze fotografické snímky analyzovat a změřené údaje zpracovat a vyhodnotit. Za zakladatele této disciplíny se pokládá francouzský vědec Aimé Laussedat (*1819 - †1907), který ji popsal již v roce 1851 ve své práci *Métrophotographie*. Fotogrammetrická metoda měření je založena na geometrických vztazích mezi předmětem a snímkem, které lze určit a následně využít k výpočtům a měřením. Aby bylo možné měřit s velkou přesností, je nutné, aby snímač měl dostatečnou rozlišovací schopnost a citlivost. Dříve se používaly metody snímkování na fotografický film na bázi bromidu stříbrného, dnes ale tuto techniku vytlačila digitální fotografie a digitální zpracování, kdy se měření neprovádí na fotografii, ale na digitálním snímku přímo v PC pomocí speciálního počítačového programu. Abychom získali digitální snímek, použijeme scaner, který převede standardně vyvolaný snímek do digitální podoby a nebo digitální kameru, která vytvoří digitální obraz přímo. Digitální kamery využívají polovodičový snímač CCD, který podle úrovně dopadajícího světla vytváří elektrický náboj, který je převeden na binární reprezentaci. Fotogrammetrie se dělí podle polohy snímače na:

leteckou: Snímač je umístěn v letadle, vrtulníku nebo satelitu a vytvořené snímky většinou slouží k vytvoření map.

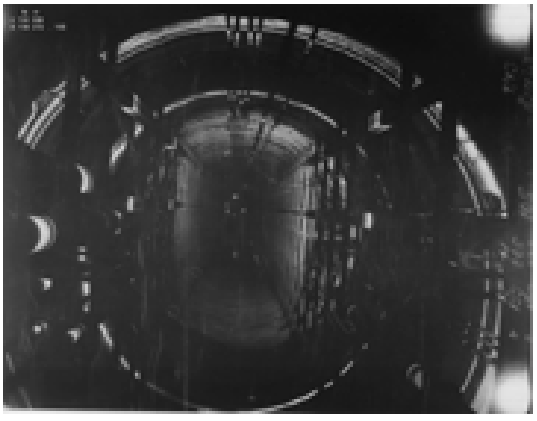
pozemní: Slouží k pozorování pohybu mostů, mapování hor, dokumentování historických budov.

Fotogrammetrie v hornictví

Mezi fotogrammetrii pozemní lze zařadit i měření důlních šachet, u kterých je nutné sledovat mimo jiné deformace profilu šachet a předem odhalit blížící se nebezpečí. Fotogrammetrie nám poskytuje bezpečný a pohodlný způsob měření v jinak nebezpečném a těžce přístupném prostředí. Na svislých jámách se provádí měření tak, že kamera, která je umístěná na těžní kleci, vytváří sérii snímků světelné stopy vytvořené na stěnách jámy pomocí *světelné roviny* (viz. obr. č. 1). Světelná rovina je vytvořena speciálním světelným zdrojem, který je zavěšen pod klecí. Dále jsou vedle klece zavěšeny olovnice, u kterých známe jejich vzdálenost a které se také zobrazí na světelné stopě a tím nám umožní určit měřítko pro přepočet mezi globálními souřadnicemi (reálné vzdálenosti) a lokálními (zachycené na snímku). Použití olovnic přináší problém způsobený jejich rozkmitáním, které způsobí špatný vypočet měřítka, a tedy i chybné měření. Na obrázku 2 je zobrazen snímek pořízený jinou metodou, u které není třeba použít zavěšených olovnic jako referenčních bodů, ale pomocí dvou rovnoběžných světelných rovin s pevně definovanou a neměnnou vzdáleností [1].



Obrázek 1: Fotogrammetrie v hornictví



Obrázek 2: Snímání pomocí dvou světelných rovin

Fotogrammetrie v lékařství

V lékařství je fotogrammetrie využívána jako neinvazivní způsob vyšetření tkání (tepen, kostí, ale i mozku). Jako snímač je velmi často používána ultrazvuková sonda, která využívá vlastnosti odrazu ultrazvuku na rozhraní tkání s různou hustotou. Podle doby odrazu ultrazvukového impulzu a jeho intenzity se následně vypočítá digitální obraz. Oproti CCD čipům je tato metoda snímání velmi zatížena šumem, jelikož intenzita ultrazvukového impulzu často nemůže být vzhledem k vyšetřované části těla příliš velká. Další problém, který se musí řešit, jsou výpadky části obrazu, a proto fotogrammetrický program musí být schopen nejen snímek filtrovat a tlumit šum, ale také umožnit doplnění chybějící části na snímku, aby bylo možné provést měření. Ultrazvuk se využívá především k vyšetření měkkých tkání, jelikož jeho shopnosti pronikat kostmi není dobrá. Pro snímání tvrdých tkání (kostí) se využívá počítačová tomografie (CT), která využívá rentgenového záření emitovaného z rentgentky, které prochází tělem a různou intenzitou dopadá na detektor. Podle naměřené intenzity se z velké série snímků vytváří obraz. Velkou výhodou CT je jeho velká přesnost a kvalita snímku. Další možností vyšetření poskytuje magnetická rezonance (MRI), která ve srovnání s CT poskytuje velmi vysoký kontrast mezi tkáněmi různé hustoty, a proto je především vhodná pro onkologická a neurologická vyšetření (rakovina, mozek), kde CT ani ultrazvuk neposkytují kvalitní snímky. Vzhledem k ceně vyšetření a dostupnosti zařízení nejsou vyšetření MRI častá.

2.2 Fotom 2008

Na katedře informatiky FEI VŠB TU Ostrava se od roku 2001 vytváří program FOTOM (ukázka na obr. č. 3), který slouží k digitálnímu zpracování snímků [1, 2]. Nejprve byl tento systém navržen pro měření důlních jam, ale postupem času se rozšířil na výkonný systém s mnoha moduly, které nabízí pokročilé možnosti detekce zájmových objektů a vizualizace měření. Současná verze *FOTOM 2008* obsahuje následující moduly:

FOTOM1 definice zájmových objektů

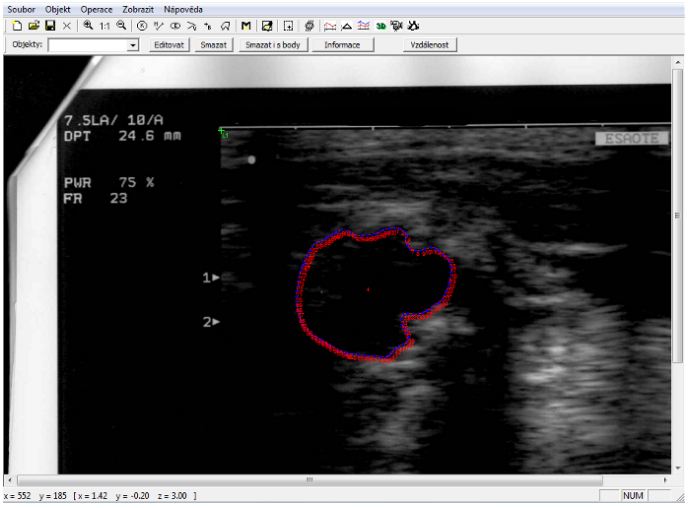
FOTOM2 2D modelování měření, porovnávání dvou měření a odchylek

FOTOM3 3D vizualizace

FOTOM4 2D animace

FOTOM5 rozpoznávání zájmových objektů

Program je vytvořen v programovacím jazyku C++. Původně byl systém vyvíjen jako jednoúčelový program a nevytvářelo se žádné veřejné API, které by mohlo využívat další moduly. Jelikož systém nemá veřejné API, není ani možné ho rozšiřovat pomocí zásuvných modulů, ale je vždy nutné nový modul zapracovat přímo do kódu programu FOTOM. Postupujícím vývojem systému je další rozšiřování stále více obtížné a vzhledem k architektuře systému je nemožné nové moduly zakomponovat přímo do systému a umožnit tak rozšiřovat a navazovat na schopnosti jiných modulů. Původní návrh programu a jeho interakce s uživatelem již nenaplňuje současné požadavky moderního systému.



Obrázek 3: Program FOTOM 2008

Proto vznikl požadavek vytvořit zcela novou generaci systému založeném na moderním jazyce a postaveném na vhodné platformě, která umožní plnou modularitu systému. Vznikne tak revoluční verze systému Fotom 2009 s nově navrženým jádrem, který díky silnému API umožní aplikaci stát se více univerzální, snadno rozšiřitelnou. Smyslem implementace nové verze systému není pouze přeprogramování původní verze systému a poskytnout (velmi velkou) výhodu modularity, ale inovativním způsobem přepracovat původní nástroje a rozšířit je o nové myšlenky způsobu definice a detekce zájmových objektů a umožnit jednodušší práci se snímkem. Současná verze systému poskytuje silné nástroje pro analýzu a vizualizaci měření, a proto nový systém ho nebude schopen zpočátku plně nahradit. Nová verze jádra proto bude provázána s Fotomem 2008 a umožní modulům export do jeho formátu.

3 Softwarové prostředky a výběr platformy

3.1 Netbeans Platform

3.1.1 Platform vs. IDE

Většina programátorů jistě zná vývojové prostředí Netbeans IDE (http://www.netbeans.org). Toto prostředí využívá stále více programátorů a spolu s Eclipse patří k nejpoužívanějšímu nástroji k vývoji programu v programovacím jazyce Java. Již podstatně méně vývojářů zná Netbeans Platform. Jaký je mezi nimi rozdíl? Netbeans IDE je vývojové prostředí usnadňující programátorům samotný vývoj. Oproti klasickým editorům, které většinou nabízí pouze zvýraznění syntaxe, Netbeans IDE poskytuje komplexní sadu nástrojů pro vývoj programů psaných v programovacím jazyce Java a to především debugger, refractor, průvodce pro složitější konstrukce, inteligentní doplňování syntaxe a samozřejmě syntaktická analýza a zvýrazňování chyb ve zdrojovém kódu. Stejně jako jeho alternativa a rival Eclipse IDE nenabízí NetBeans prostředky pro vývoj programů čistě v jazyce Java, ale umožňuje plnohodnotně vytvářet programy v jazyce PHP, Ruby, C++ atd. Tím ale jeho síla nekončí! Díky otevřenosti a modularitě vznikají další pluginy rozšiřující základní možnosti vývojového prostředí, a tak lze jednoduše nainstalovat editor UML, databáze MySQL nebo integrovat framework Struts pro vývoj JEE aplikací.

Všechny tyto možnosti, které IDE má, jsou dány základem, na kterém je prostředí postaveno a tím je právě Netbeans Platform (http://platform.netbeans.org). Jak již název napovídá, Netbeans Platform je framework pro vývoj *modulárních aplikací*, poskytuje velké množství API pro snadnou implementaci základních částí, které potřebuje každá aplikace, jakou je manažer oken, panely a ikony, reakce a zpracování událostí. Vedle těchto základních schopností poskytuje nástroje pro mnohem důmyslnější mechanizmy, jako je modularita, komunikace mezi moduly a závislosti mezi nimi, aktualizace atd.

Praftorma NetBeans tedy tvoří základní kámen pro vývoj moderních modulárních systémů, poskytuje velké množství programových prostředků pro správu oken, akcí a modulů, které ulehčují práci programátora a umožňují mu zaměřit se na řešený problém. Síla platformy spočívá v jasném API, které lze snadno pochopit a které umožňuje již po krátké chvíli psát vlastní moduly, velké komunitě vývojářů a dostupnosti literatury a zdrojů online. Nespornou výhodou jsou také pravidelné aktualizace platformy.

Vedle Netbeans Platform existuje (mimo jiné) alternativa také s početnou komunitou a to Eclipse RCP (http://www.eclipse.org/platform). Z velké části se jedná o podobné produkty, jelikož oba řeší podobný problém. Rozhodování mezi platformou NetBeans a Eclipse z velké části není otázka porovnání jejich schopností, jelikož jejich síla a schopnosti jsou vyrovnány. I když existuje několik rozdílů (často uváděný zásadní rozdíl mezi platformami je použití AWT platformou Eclipse, zatímco Netbeans Platform využívá Swing) z velké míry často rozhoduje i sympatie k jednotlivým produktům a také v jakém vývojovém prostředí programátor rád pracuje.

V následujícím textu popíši vybrané části NetBeans Platform verze 6.5.1, které jsou nejvíce používané a které jsou později použity v návrhu a implementaci systému.

3.1.2 Module System API

Nejdůležitější částí NetBeans Platform je *Module System API*, který programátorovi umožňuje stavět aplikace ze samostatných částí - modulů. Díky modulům se aplikace může velmi snadno rozšiřovat, měnit, ale i aktualizovat a distribuovat. Modul je samostatný logický prvek, řešící optimálně jednu věc. Může to být pouhé přidání položky do menu a nebo kompletní sada nových funkcí. Netbeans Platform poskytuje nástroje pro tvorbu modulů, jejich instalaci a také programátorovi dává nástroje pro komunikaci mezi moduly. Správce rozšíření je automaticky integrován do aplikace, která platformu využívá a díky němu se aplikace snadno rozšiřuje a aktualizuje. Více [8].

Manifest Každý modul je distribuován zvlášť v JAR souboru, který mimo jiné obsahuje *manifest* soubor. Tento soubor určuje základní informace o modulu jako je jeho jméno, popis, třídu řídící případnou instalaci modulu, module-layer a verzi. Tyto informace slouží k identifikování modulu, definici závislostí atd. Ukázka manifest souboru je na výpisu 1.

```
Manifest-Version: 1.0
OpenIDE-Module: org.fotomapp.basictools
OpenIDE-Module-Install: org/fotomapp/basictools/Install.class
OpenIDE-Module-Layer: org/fotomapp/basictools/layer.xml
OpenIDE-Module-Localizing-Bundle: org/fotomapp/basictools/Bundle.properties
OpenIDE-Module-Specification-Version: 1.0
```

Výpis 1: Ukázka manifest souboru

Instalace V manifest souboru položka *OpenIDE-Module-Install* určuje třídu, která obsluhuje funkce při instalaci (metoda *installed)*, odinstalaci (metoda *uninstalled)*, aktualizaci (metoda *updated)*, uzavření modulu (metoda *closed)* a nebo může obsluhovat požadavek na validaci (metoda *validate)*, která je vhodná například na kontrolu licenčního klíče. Třídu napíšeme velmi snadno, a to děděním třídy **ModuleInstall** a implementací požadovaných metod. V zásadě se ale **nedoporučuje** tuto třídu používat, ale místo ní použít module-layer, viz dále.

Module-Layer Odkaz na soubor module-layer je specifikován v manifestu na řádku *OpenIDE-Module-Layer*. Struktura souboru je velmi podobná struktuře standardních XML souborů. Root je definován značkou <*filesystem*>, dále jsou používány značky pro adresář <*folder*> a soubor <*file*>. Pro tyto značky je možno dále specifikovat atributy pomocí <*atr*>. Používání module-layer má velkou výhodu v tom, že bez psaní kódu lze definovat velké množství funkcí platformy:

- definovat menu a akce
- definovat nástrojové panely
- klávesové zkratky
- vytvořit paletu nástrojů

- definovat rozmístění oken

Pomocí module-layer, který je většinou definován v souboru *layer.xml*, lze deklarativním způsobem vytvářet virtuální souborový systém, který se spuštěním modulu transformuje do fyzického souborového systému v pracovní složce uživatele aplikace. Takto lze tedy definovat například i pracovní adresáře, které jsou dále využívány pro perzistenci aplikace. Platforma umožňuje poslouchat změny v takto definovaném souborovém systému, a proto je možné instalovat nové moduly za běhu programu a reagovat na změny v souborovém systému.

Verze a závislosti *Verze a závislosti* poskytují možnost jak zamezit instalaci nekompatibilních modulů, modulů, které mají závislost na novější verze instalovaných modulů atd. Aby byla kontrola závislostí a verzí možná, je nejprve nutné určit unikátní název (ID) modulu pomocí tagu *OpenIDE-Module* v manifestu. Takto určíme jednoznačný identifikátor modulu, který nesmíme už měnit. Dále modul o sobě musí definovat svou verzi značkou *OpenIDE-Module-Specification-Version* (případně i OpenIDE-Module-Implementation-Version). Pro definici modulů, na kterých je modul závislý, použijeme značku *OpenIDE-Module-Module-Dependencies*. Takto definovaný modul lze snadno rozšiřovat a vytvářet jeho nové verze, které systém může automaticky kontrolovat. NetBeans IDE poskytuje snadný průvodce pro vytvoření distribucí nových verzí modulů (soubory **mbm**), které lze samostatně instalovat a nebo automaticky aktualizovat například z internetu.

3.1.3 Lookup API

Lookup je mechanizmus pro hledání instancí objektů [10]. Můžeme si jej představit jako strukturu *Map*, kde klíčem je objekt *Class* (MojeTrida.class) a hodnotou jsou instance této třídy. Často používaný je **globální lookup**, který slouží k hledání objektů registrovaných v celém systému. Globální lookup připomíná *java.util.ServiceLoader* (v JDK od verze 1.6), umožňuje tedy získat instanci třídy (service provider) poskytující nějakou funkčnost, ke které známe rozhraní. V mnoha ohledech je ale použití globálního lookupu [3] vhodnější:

- lookup podporuje posluchače
- lookup není finální a lze jej rozšířit
- funguje i ve verzích Javy starších než 1.6

Pokud chceme registrovat do aplikace službu tak, aby se dala pomocí globálního lookupu najít, pak vložíme do adresáře *META-INF/services* soubor se stejným jménem jako je rozhraní služby a jeho obsahem bude umístění třídy, která toto rozhraní implementuje. Například implementujeme službu *sluzby.MojeSluzba*, pak stačí vytvořit soubor *META-INF/services/sluzby.MojeSluzba* a do něj umístit jediný řádek definující, kde global lookup nalezne implementaci, např. *org.MojeSluzbaImpl*. Ve verzi platformy 7.0, která při psaní této práce je dostupná pouze jako beta verze, bude možno používat navíc anotaci *@ServiceProvider*. Nalezení poskytovatele služby a jeho použití je pak velmi jednoduché (výpis 2).

```
MojeSluzba s : Lookup.getDefault().lookup(MojeSluzba.class);
s.akce();
```

Výpis 2: Vyhledání služby pomocí globálního lookupu

Vedle vyhledání služeb slouží lookup i jako jakási úschovna objektů, které patří k objektu. Mnoho objektů mají metodu *getLookup()* a umožňují tak získat přidružené objekty. Velmi používaná je metoda *Utilities.actionsGlobalContext()*;, která vrací lookup podle **aktuálního kontextu**. V základní implementaci kontextu v NetBeans se kontext mění podle aktivního okna (TopComponent viz. 3.1.4) a nebo aktivního Node objektu (viz. 3.1.5). Pokud ale takové chování nevyhovuje povaze programu, je možné implementovat vlastní kontext (viz. 4.1.3). Základní využití kontextového lookupu může vypadat následovně:

- V modulu API je definováno rozhraní objektu, který slouží jako obalovací objekt obrázku a definuje na něm další operace.
- Modul obsahující funkci pro zobrazení obrázku tento objekt získá (například mu ho předá jiný modul implementující perzistenci), zobrazí a umístí do svého lookupu. Aktivní okno (vždy TopComponent) definuje kontext aplikace, a proto i právě zobrazovaný objekt je přístupný pomocí kontextového lookupu.
- Jiný modul, který chce jakkoliv upravit obrázek, získá lookup kontextu a v něm vyhledá objekt podle jména třídy, kterou objekt implementuje. Tento modul nemusí nic vědet o tom, který modul se stará o zobrazování a jak to dělá. Získaný objekt může libovolně zpracovat a nebo zobrazit jiným způsobem (například vytvořit náhled na obrázek).

Jak již bylo řečeno, velkou výhodou lookupu (globálního, kontextového nebo lookupu, který nabízí jednotlivé objekty) je možnost zaregistrovat posluchače (LookupListener). Vrátím-li se k předchozímu modelu použití kontextového lookupu, pak je možné například poslouchat kontextový lookup a kontrolovat, zda se v něm neobjevil obrázek a podle toho reagovat. Jenom je třeba si uvědomit, že posloucháme změnu lookupu a ne změny jednotlivých objektů v něm, pokud tedy někdo upraví vlastnosti objektu, který je umístěný v lookupu, pak se posluchači lookupu o ničem nedozví. K tomuto slouží *PropertyChangeListener*. Pokud bychom i přesto chtěli upozornit posluchače na změnu objektu bez použití *PropertyChangeListener*, nabízí se možnost (která je často doporučovaná) jednoduše objekt z lookupu odstranit a ihned vrátit zpátky. Více o lookupu viz. [9, 10, 12].

3.1.4 Window System API

Při vytváření oken v aplikaci postavené na platformě NetBeans nelze používat standardní třídy používané ve Swing. NetBeans obsahuje vlastní manažer oken, který se stará o dokování, maximalizaci, přesunování a upravování velikosti oken. Aby se nové okno mohlo začlenit do okenního manažeru, musí být odvozeno od třídy *TopComponent*

[13]. V podstatě veškeré viditelné panely jsou odvozeny z této třídy. NetBeans IDE nabízí průvodce, který vytvoří veškeré potřebné třídy a zaregistruje vytvořenou komponentu v layer.xml (3.1.2), ve kterém definujeme, na které pozici bude komponenta zadokována. Základní možnosti jsou:

editor hlavní okno

explorer levý panel

output spodní panel

Vedle těchto základních umístění zadokování okna lze definovat pozice další a vytvořit si tak například pozici „navigator“ (levý dolní roh aplikace), která není přímo podporována. Třída TopComponent nejen velmi ulehčuje programátorovi práci s okny, ale nabízí také možnost definice vlastních akcí (viz. 3.1.7), může propagovat undo/redo manažera (v aplikaci znázorněn šipkami zpět, vpřed), který umožňuje ukládat akce provedené uživatelem a následně je vracet a znovu provádět a v neposlední řadě ovládá a aktivuje Nodes a definuje tak kontext lookup. Dále TopComponent může zobrazit svou paletu nástrojů, která je definována v layer.xml a reagovat na výběr nástrojů.

3.1.5 Nodes API

Node [14] poskytuje nástroje pro vizuální reprezentaci objektů. Může reprezentovat například soubory a adresáře ze souborového systému (DataNode, DataFolder) nebo jakékoli jiné objekty. Node tvoří základní kámen pro reprezentaci objektu a jeho vlastností, jsou to ale pouze obalovací objekty, samy žádná data nenesou. Díky *Explorer API* můžeme kolekce objektů Node zobrazovat různými způsoby, jako jsou stromová menu, seznamy atd.

Pokud chceme vytvořit vlastní objekty Node, můžeme využít třídu *AbstractNode* (není třídou abstraktní) případně některou z odvozených tříd (např. BeanNode, která je vhodná pokud máme JavaBean objekt a chceme ho graficky reprezentovat). Přepisem metod třídy AbstractNode můžeme zobrazovat vlastnosti v komponentě PropertyEditor (editor vlastností), ovlivníme styl zobrazování jména objektu v seznamech, poskytneme seznam akcí, které se nabídnou při kliku myší atd.

3.1.6 Vlastnosti

Node, lépe řečeno objekt zabalen v objektu Node, vetšinou chce zobrazovat své vlastnosti. K tomu Node poskytuje metodu *createSheet()*, která je volána, když se Node stane aktivním (který node je aktivní obvykle určuje objekt třídy TopComponent). Vlastnosti vrácené touto metodou jsou zobrazovány v editoru vlastností (property editor), obvykle omístěn v pravém dolním rohu aplikace. Pro zobrazení vlastností slouží třída *Property*, která vytvoří editor pro určené proměnné a sama podle předaného typu zvolí vhodný editor (pro objekty Color.class vytvoří paletu barev, String.class vytvoří vstupní pole atd.), případně pokud nám editor nevyhovuje, můžeme si vytvořit vlastní (např. pro editaci datumu). Tyto jednotlivé vlastnosti se shlukují v objektu typu *PropertySet* podle společných vlastností (např. všechny vlastnosti týkající se barev sloučíme dohromady)

a vzniknou tak celky, které lze v editoru přepínat. Objekt *Sheet* nám dovoluje vytvářet různé druhy vlastností, které mohou být rozděleny například podle úrovně uživatelských schopností.

3.1.7 Actions API

Akce [15] jsou třídy, které jsou volány při interakci uživatele se systémem. Může to být klik v menu, nástrojové listy nebo vyvolání klávesové zkratky. Pokud chceme přiřadit objektu Node nějaké akce, není třeba používat Actions API, ale vystačíme si s možnosti Swing. Actions API využijeme pokud potřebujeme akce závislé na kontextu nebo používat systémové akce (Save, Open).

Akce závislé na kontextu Pokud existuje akce, která mění své chování v závislosti na aktuálním kontextu aplikace, pak použijeme *CallbackSystemAction*. U této akce můžeme nastavovat *ActionPerformer* podle aktuálního stavu aplikace a tím ovlivňovat chování akce. Další vhodnou třídou, kterou ve své akci můžeme rozšířit, je *NodeAction*. Použitím této třídy jsme schopni povolit a zakázat provedení akce podle aktuálně aktivního Node.

Instalace akce Instalace vytvořené akce je velmi jednoduchá a opět se deklaruje v souboru layer.xml (výpis 3). V něm si jednouduše definujeme samotnou akci, umístění v menu případně v nástrojovém panelu, ikonu, klávesovou zkratku.

```
<filesystem>
    <folder name="Menu">
        <folder name="Nastroje">
            <file name="org-cesta-MojeAkce.instance">
                <attr name="position" intvalue="100"/>
            </file>
        </folder>
    </folder>
    <folder name="Shortcuts">
        <file name="S-A-Left.shadow">
            <attr name="originalFile" stringvalue="Actions/Nastroje/org-cesta-MojeAkce.
                instance"/>
        </file>
    </folder>
</filesystem>
```

Výpis 3: Instalace akce a klávesové zkratky do systému

3.1.8 Nastavení

Platforma NetBeans umožňuje centralizovanou správu nastavení aplikace pomocí *Options Window*. Pokud chce programátor modulu umožnit uživateli nastavení parametrů jeho modulu, může využít průvodce ve vývojovém prostředí Netbeans, který vytvoří všechny potřebné třídy a zaregistruje nový panel nastavení do layer.xml. Tímto se vytvoří třída, která dědí z třídy *OptionsPanelController* a třída samotného panelu. Pomocí

návrháře GUI jsme schopni snadno vytvořit požadovaný panel, který pro uložení a načtení nastavení poskytuje metody *save()* a *load()*.

3.2 Java Advanced Imaging

Použití jazyka Java pro vývoj aplikací, která bude pracovat převážně s obrázky, sebou přináší problém přístupu k obrázku a jeho zpracování. V balíku Java2D poskytuje Java základní možnosti zpracování obrazu, jako je rozmazávání, ostření, geometrické transformace. Možnosti tohoto balíku jsou ale velmi omezené a obzvláště svým výkonem nejsou vhodné k náročnějšímu zpracování složitými filtry a transformacemi. Java Advanced Imaging (JAI) API rozšiřuje možnosti platformy Java a umožňuje sofistikované a vysoce výkonné zpracování obrazu a poskytuje třídy, které vysoce převyšují možnosti balíku Java2D [6].

3.2.1 Výkon

JAI bylo vytvořeno jako API splňující požadavky většiny druhů aplikací, které zpracovávájí obrazy. Jeho API je snadno rozšiřitelné a umožňuje imaplementaci dalších operátorů. Většina API pro práci s obrazy jsou vázána na konkrétní platformu, pro kterou byly vytvořeny, ale JAI poskytuje platformově nezávislé rozhraní, které je plně funkční na jakékoli platformě s JVM. Tato vlastnost by se mohla zdát jako nevýhoda vzhledem k výkonu aplikace, ale není tomu tak. JAI je i přes svou platformovou nezávislost velmi výkonné, jelikož existuje několik implementací API vytvořené pro konkrétní platformy, a proto JAI může využívat plnou hardwarovou akceleraci a schopnosti konkrétních platforem (například MMX u procesorů Intel). Pokud pro platformu, na které chceme využívat schopností JAI, neexistuje konkrétní implementace, pak existuje i platformově nezávislá implementace distribuovaná jako archiv JAR, která ovšem samozřejmě neposkytuje takový výkon.

3.2.2 Možnosti JAI

Mezi největší přednosti JAI patří podpora velkého množství formátu obrázku, které lze snadno zpracovávat. Existuje přes 100 operátorů, které umožňují většinu používaných operací při zpracování obrazu. Vedle základních operátorů pro zaostření, rozmazání, detekci hran, logické operace a geometrické transformace poskytuje JAI nástroje k definování oblastí zájmu, vytvoření a zpracování histogramu, statistické operátory a operátory pracující ve frekvenční doméně (více [6, 7]). Ve výpisu zdrojového kódu č. 4 je předveden způsob práce s JAI při řešení problému rotace obrazu o úhel „uhelRotace“ podle bodu „stredRotace“.

```
PlanarImage image = JAI.create("fileload",  "obrazek.bmp");
ParameterBlock pb = new ParameterBlock();
pb.addSource(image);
pb.add(stredRotaceX);
pb.add(stredRotaceY);
pb.add(uhelRotace);
```

```
pb.add(new InterpolationBilinear());
PlanarImage rotace = JAI.create("rotate", pb);
```

Výpis 4: Ukázka použití JAI - rotace

4 Návrh systému Fotom 2009

Návrh jádra systému a jeho modulů vychází z praktických zkušeností získaných používáním programu Fotom 2008. Analýzou funkcí a nedostatků aplikace vznikne specifikace požadavků na architekturu systému a jeho moduly. Jádro systému bude navrženo jako snadno rozšiřitelné, univerzální prostředí pro rozšiřující moduly a jeho analýza a implementace bude probíhat ve spolupráci s Bc. Janem Králem. Moduly pro definici objektů, měření a skicování budou navrženy tak, aby svou funkčností zastoupily a inovativním přístupem vylepšily funkčnost programu FOTOM1, ale také novými nástroji umožnily snadnější definici objektů a úpravy snímků.

4.1 Návrh a realizace jádra systému

4.1.1 Byznys modely

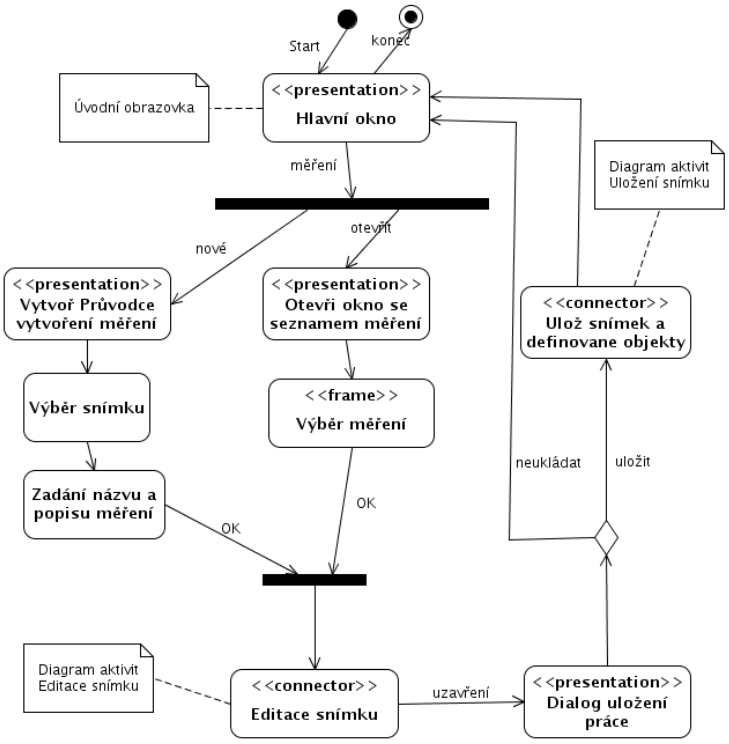
Diagramy aktivit a doménový model Na obrázku 4 je zobrazen diagram aktivit aplikace jako celku. Jeho hlavní úkol je zobrazit, jak by měly pracovat hlavní části aplikace. Dále diagramy na obrázku 5 a 6 zobrazují sekvence aktivit a interakce s GUI při editaci snímku a jeho ukládání. Doménový model je znázorněn na obrázku 7.

4.1.2 Specifikace požadavků

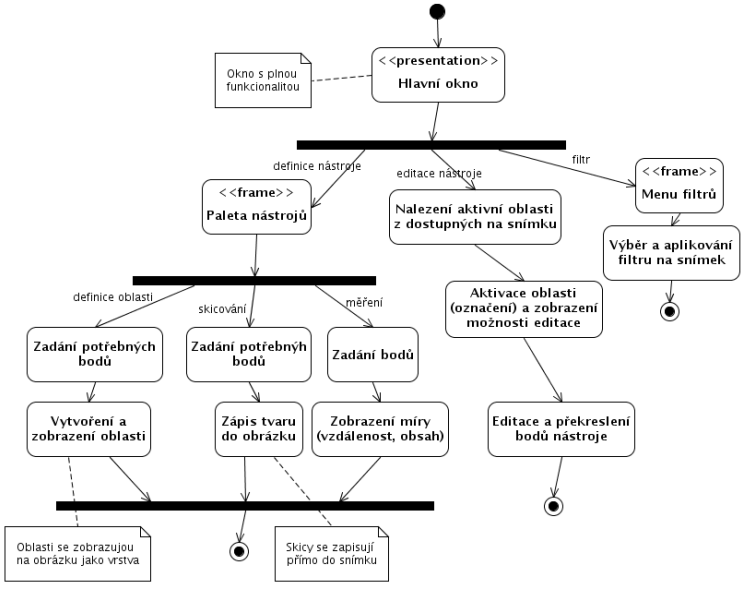
Hlavním úkolem je vytvořit jádro nové verze aplikace FOTOM 2008. Současný stav aplikace je již nevyhovující a aplikace se obtížně rozšiřuje. Dalším problémem je silná platformová závislost a prakticky není možné implementovat nezávislé moduly, jelikož neexistuje společné API. Současná aplikace je napsána v programovacím jazyce C++ a nemá programovou dokumentaci, jsou pouze dostupné zdrojové kódy.

Základním požadavkem je umožnit snadnou rozšiřitelnost aplikace bez nutnosti zasahovat do samotného jádra aplikace. Program by měl být platformově nezávislý. Aplikace by měla mít okno, ve kterém bude seznam jednotlivých měření, a kde jednotlivá měření půjdou strukturovat do adresářů. Dále okno pro editaci samotných snímků pomocí nástrojů, okno zobrazující dostupné nástroje pro editaci snímku a okno zobrazující vlastnosti vybraného snímku nebo nástroje.

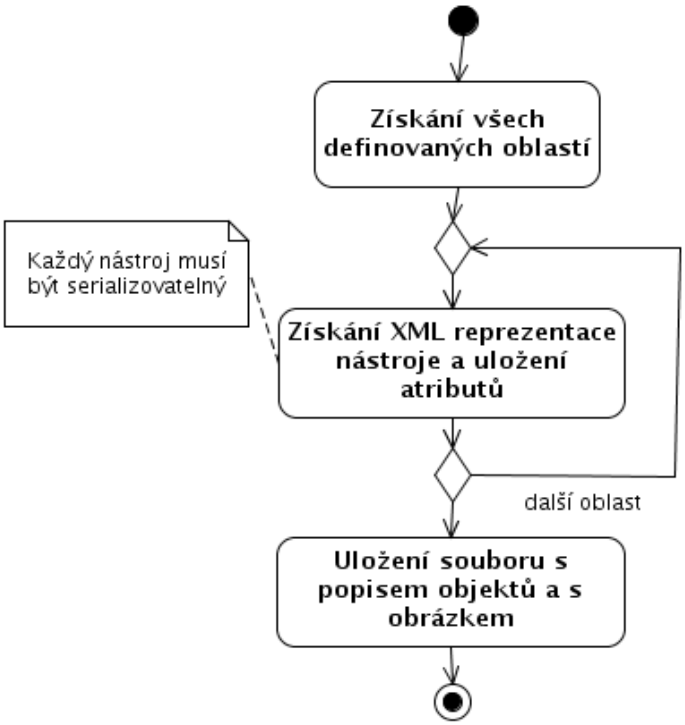
Aplikace musí mít možnost snadného rozšiřování, proto je nutné vytvořit veřejné API a definovat vhodná rozhraní, která umožní modulům jednoduše přistupovat k právě zpracovávanému snímku, upravovat ho a získávat z něj informace, které může vizualizovat přímo na editovaném snímku. Dále je potřeba popsat rozhraní nástrojů pro definici objektů, které slouží k vymezení hranic zájmových objektů na snímku, které chceme dále sledovat (měřit mezi nimi vzdálenost, pozorovat změnu velikosti objektu v čase atd.). Vedle nástrojů pro definici měření je potřeba umožnit vytvářet nástroje schopné editovat snímek, které budou sloužit k úpravám méně kvalitního snímku a také jako filtry pro zvýraznění hranice mezi hledanými objekty a pozadím.



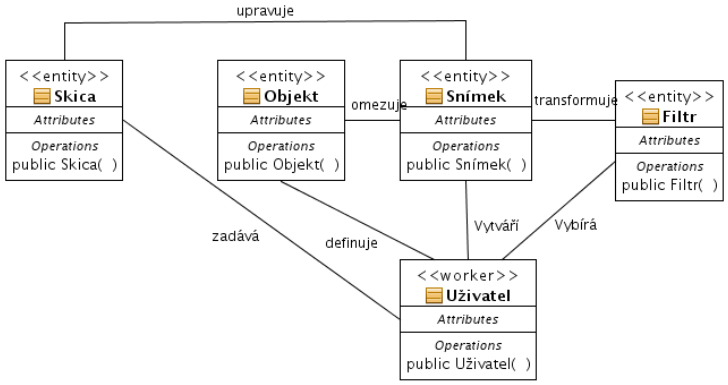
Obrázek 4: Diagram aktivit aplikace



Obrázek 5: Diagram aktivit editace snímku



Obrázek 6: Diagram aktivit uložení



Obrázek 7: Doménový model

Nástroje Samotná realizace jednotlivých nástrojů bude implementována v zásuvných modulech, které budou používat společné jádro aplikace a jeho API. Jádro bude vytvořeno tak, aby podporovalo následující nástroje a operace:

Nástroje pro definování oblastí budou umožňovat manuální nebo automatickou definici objektů. Definice bude probíhat označením hranice zájmového objektu pomocí myši. Označená hranice zvýrazněna čárou zvolené barvy, případně vyplněna barvou. Označení objektu neznamená zásah do samotného snímku a definované oblasti se budou zobrazovat na snímku jako další vrstva. Nástrojům, které budou kompatibilní s verzí systému Fotom 2008, bude umožněno exportovat se do jeho formátu.

Nástroje pro skicování a filtry budou sloužit k úpravám snímků, které vinou špatné kvality snímače, zhoršenými podmínkami a nebo zvolenou technikou jsou jakkoli poškozeny a je třeba je upravit tak, aby odpovídaly realitě. To především znamená dokreslit chybějící hranice sledovaných objektů a umožnit tak správně definovat jejich hranici a umožnit měření. Velmi často snímky obsahují šum nebo nejasné hranice mezi objekty a pozadím. Tyto vady snímku budou řešit filtry schopné šum zmírnit a pomocí prahování a detekcí hran hranici objektů zvýraznit. Veškeré úpravy snímku se musí provádět na kopii, aby se originální snímek nezměnil a aby bylo možné se při příštím měření zaměřit na jiné objekty, které mohly být aplikací nevhodných filtrů a dodatečnými úpravami znehodnoceny.

nástroje pro měření by měly umožnit rychlé odměření vzdálenosti mezi objekty na snímku. Je třeba navrhnout rozhraní pro nástroj, který bude definovat **lícovací body**, aby bylo možné měřit vzdálenost ve skutečných jednotkách a ne pouze v obrazových bodech. Tyto body slouží k definici většinou dvojici bodů na snímku, u kterých je známo jejich *globální* umístění (definují tak pozici snímku v realitě vzhledem k ostatním snímkům) a jejich reálná vzdálenost (definují tak měřítko).

Implementací výše uvedených rozhraní vznikne celá řada nástrojů, které budou schopny definovat různé typy objektů na snímku a nebo snímek upravovat. Aby byly nástroje uživateli vhodně přístupné, bude vytvořen panel nástrojů, který je bude zobrazovat. Každý nástroj musí být schopen vytvořit svou kopii, aby bylo umožněno importovat definované objekty do podobného snímku a dále každému definovanému objektu bude umožněna adaptace na nový snímek, tedy pokusit se vyhledat novou hranici objektu na novém snímku.

Perzistence Nedílnou součástí každého programu je perzistence, a proto i nově navrhovaný systém musí být schopen uložit definované objekty do souboru a to nejlépe ve formátu XML. Volbou XML souboru umožníme snadnou editaci definovaných objektů v textovém editoru a umožníme ostatním případným aplikacím jednoduše soubor importovat. S definovanými objekty je vhodné uložit do pracovního adresáře i samotný snímek, pak měření nebude závislé na originálním snímku, který se dále může zpracovávat zcela jiným způsobem.

Prezentace měření Zobrazením snímku na plátně se umožní jeho editace a měření, proto se musí vytvořit mechanismus, který umožní modulům být imformován o nově editovaném snímku a budou moci získat na něm definované objekty a reagovat na aktuálně zvolený nástroj. Moduly musí mít možnost jak zobrazit svou informaci na snímku, a proto se musí vytvořit jednotný přístup ke kreslicímu plátnu. Každé provedené měření bude zobrazeno v panelu, kde bude možno jednotlivá měření strukturovat do složek a vytvářet nová měření. Panel bude poskytovat akce pro otevření měření v oknu pro definici objektů a také rozhraní pro definování dalších akcí pro různé způsoby otevření snímku. Mohou tak vzniknout další moduly umožňující zobrazit statistické údaje ze snímku, poskytující kalibrační nástroje nebo okno vytvářející 3D pohled na sérii snímků.

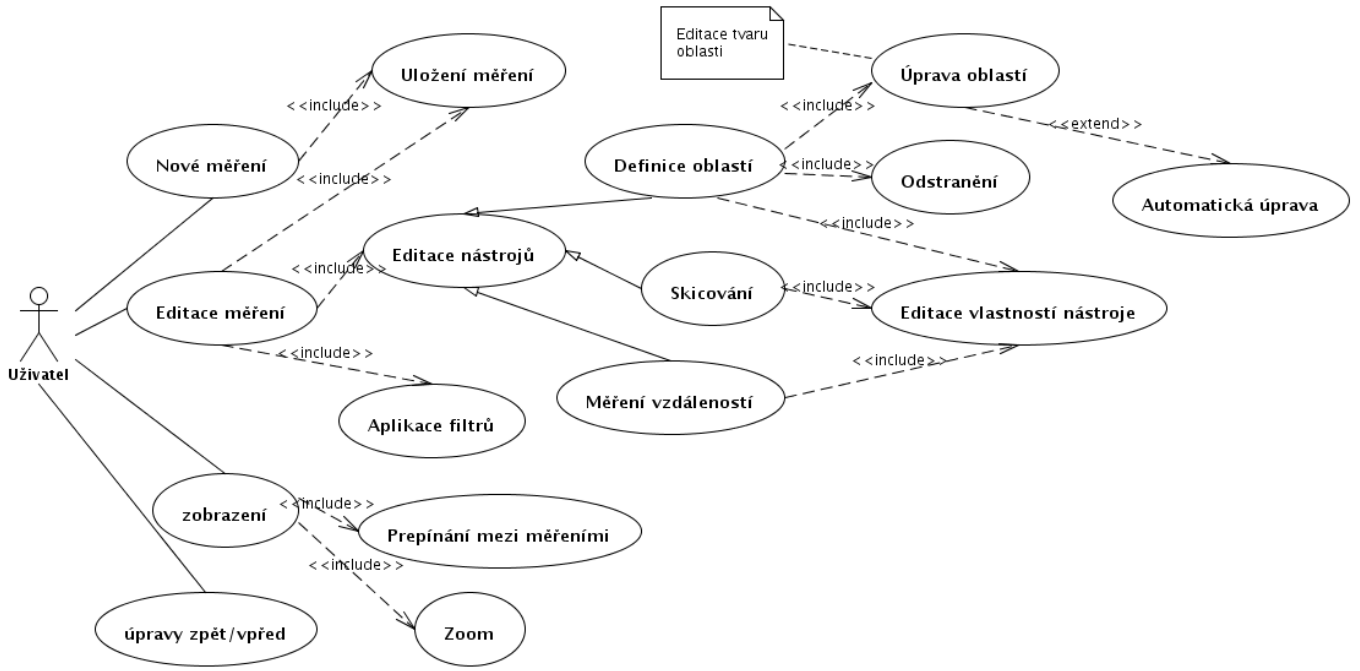
Další funkce systému Pro usnadnění práce s programem je vhodné, aby aplikace umožňovala operace *zpět* a *vpřed* s dostatečným počtem kroků. Systém si bude pamatovat každou změnu v definici objektů na snímku a dále jakoukoliv úpravu samotného snímku. To znamená, že každý nástroj musí být schopen vytvořit událost kroku zpět a vpřed. Celý systém by měl být snadno aktualizovatelný, nejlépe automaticky z internetu. Pro základní okno editace snímku je vhodné umožnit zobrazit jak originální snímek, tak snímek upravený skicováním. Samozřejmostí je poskytnutí funkce přiblížení a oddálení snímku. Je potřeba, aby byl systém propojen s existující verzí programu Fotom 2008, a proto se umožní spouštění jednotlivých jeho modulů z menu. Aplikace Fotom 2008 je napsána pouze pro operační systém Windows, a proto tato funkce bude vázána pouze na tento operační systém.

Diagramy případů užití Obrázek 8 zobrazuje diagram případů užití aplikace. Jedná se o globální náhled na všechny funkce, které budou podrobněji popsány písemně.

Případ užití **nové měření** popisuje založení nového měření z existujícího snímku. Existující snímek bude průvodcem importován do programu a dále se bude pracovat s touto kopií snímku. K novému snímku budou přidány základní informace jako je název nebo popis a nové měření bude uloženo do souboru.

Editace měření zahrnuje případy užití, které popisují práci s objekty na snímku. Základní případ užití je **definice oblastí**, který je v diagramu na obrázku 8, zahrnuje *úpravy oblastí*, tedy samotnou definici objektu a jeho pozdější korekci, odstranění objektu a editaci vlastností (barva hranice, styl a způsob zobrazování, chování při detekci objektu). Veškeré tyto případy užití předpokládají, že je vybrána jedna oblast na snímku. Pokud je oblastí definovaných na snímku více (mohou se i překrývat), je aktivní objekt vybrán podle jeho obsahu, kdy se preferuje výběr menší oblasti. Postup výběru je znázorněn na obrázku 4.1.2 sekvenčním diagramem.

Případ užití **aplikace filtrů** ukazuje použití zásuvných modulů, které získají právě editovaný snímek a na něj aplikují filtry nebo transformace. Příkladem může být výběr filtru pro provedení detekce hran. Filtr získá pracovní kopii snímku (originální snímek se nesmí měnit), aplikuje masky pro detekci hran a výsledný obraz zobrazí. Takto upravený



Obrázek 8: Use case aplikace

snímek je připraven k dalším úpravám a nebo hledání a měření objektů.

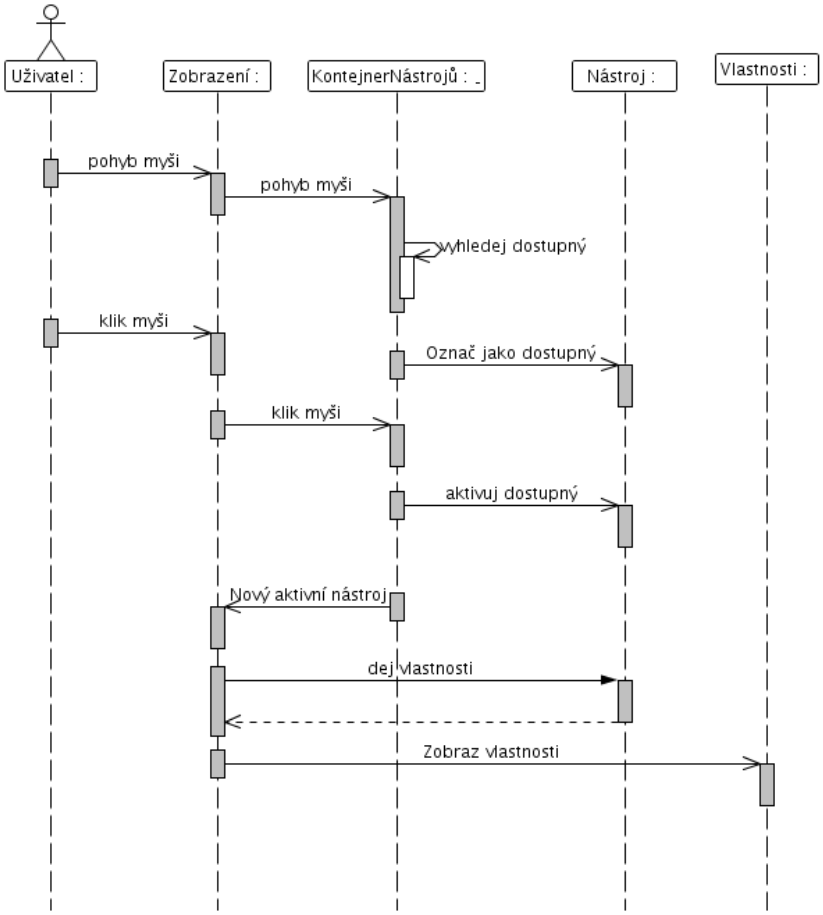
Zobrazení zahrnuje zobrazování snímku a vykreslování zadaných nástrojů pro definici oblastí. Vedle zobrazování snímku a nástrojů musí být systém schopen reagovat na požadavek překreslení části změněného snímku. Dále zahrnuje možnost přibližování snímku a oddalování, kdy při požadavku změny přiblížení se překreslí snímek a přepočítájí se souřadnice tak, aby definované objekty nemusely počítat s přiblížením. Jelikož je umožněno editovat a zobrazit více snímku najednou, zobrazení řídí i přepínání mezi jednotlivými snímky.

Sekvenční diagramy Následující diagramy zobrazují propojení jednotlivých částí programu a interakce mezi sebou a uživatelem. Obrázek 9 zobrazuje postup výběru již vytvořeného nástroje, který definuje oblast. Objekt *Uživatel* reprezentuje člověka, který pomocí myši chce označit nástroj. Označení probíhá pohybem kurzoru myši na objektu, který reprezentuje *zobrazení*, tedy panel, ve kterém je zobrazen snímek a veškeré nástroje. Akce myši je rozeslána všem nástrojům, které jsou na snímku definovány a následně je určen *dostupný* objekt, což je ten, do kterého ukazuje myš (pokud ukazuje do více nástroju, pak se vybere ten s nejmenším obsahem). Při kliku myší se vybraný nástroj aktivuje tak, že se mu pošle zpráva, že byl aktivován a získají se jeho vlastnosti, které lze zobrazit v *panelu pro editaci vlastností*.

4.1.3 Analýza a návrh

Kontextový lookup V části (3.1.3) byla popsána struktura lookup, která je poskytována platformou jako nástroj pro komunikaci mezi moduly, ale i mezi objekty v rámci modulu. Jedná se o velmi univerzální nástroj, který lze bez zásahu do jeho struktury využívat v mnoha druzích aplikací. Základní implementace poskytování *kontextového lookupu* není vhodná přímo pro tento systém. Původně je ContextGlobalProvider (rozhraní pro poskytování lookupu) implementováno ve *window system* (viz. 3.1.4) a to tak, že Lookup právě aktivní TopComponenty je delegován jako kontextový. V této aplikaci ale chceme, aby každý modul mohl delegovat svůj Lookup bez ohledu na to, jestli jeho prezenční vrstva je aktivní (modul ve své podstatě ani nemusí mít žádnou TopComponentu). Hlavní výhodou tohoto řešení bude, že právě aktivní snímek bude neustále přístupný, i když nebude momentálně aktivní TopComponenta, která snímek zobrazuje. Jednotlivé moduly budou moci poskytnout svůj lookup a v API implementující rozhraní ContextGlobalProvider se tyto lookup objekty získají a delegují jako jeden objekt. Návrh je popsán obrázkem 10.

Perzistence Pro perzistenci definovaných objektů byl zvolen přístup serializace objektů do XML. Tento způsob přináší výhody snadného uložení objektu. Jedinou podmínkou, kterou musí nástroj splňovat je, že musí být JavaBean. To v důsledku pouze znamená, že musí mít definován konstruktor bez parametrů a pro veškeré proměnné, které mají být uloženy, musí definovat veřejné přístupové metody (více [11]). Nástrojům často pouhá



Obrázek 9: Sekvenční diagram výběru objektu

deserializace objektu při načítání měření ze souboru nestačí a potřebují inicializovat vnitřní struktury podle uložených parametrů. Proto je v API definována metoda *init()*, která se volá vždy při zobrazení měření. V této metodě může každý nástroj provést inicializaci, jelikož má zaručeno, že veškeré uložené parametry jsou nastaveny.

Diagram tříd Následující diagramy zobrazují statický pohled na jádro systému. Vzhledem k velikosti tříd a počtu metod jsou diagramy zjednodušeny a nejsou zobrazeny například metody, které zpracovávají události myši nebo poslouchání událostí. Je pouze naznačeno, že třída implementuje dané rozhraní, ze kterého již vyplývá, které metody musí implementovat.

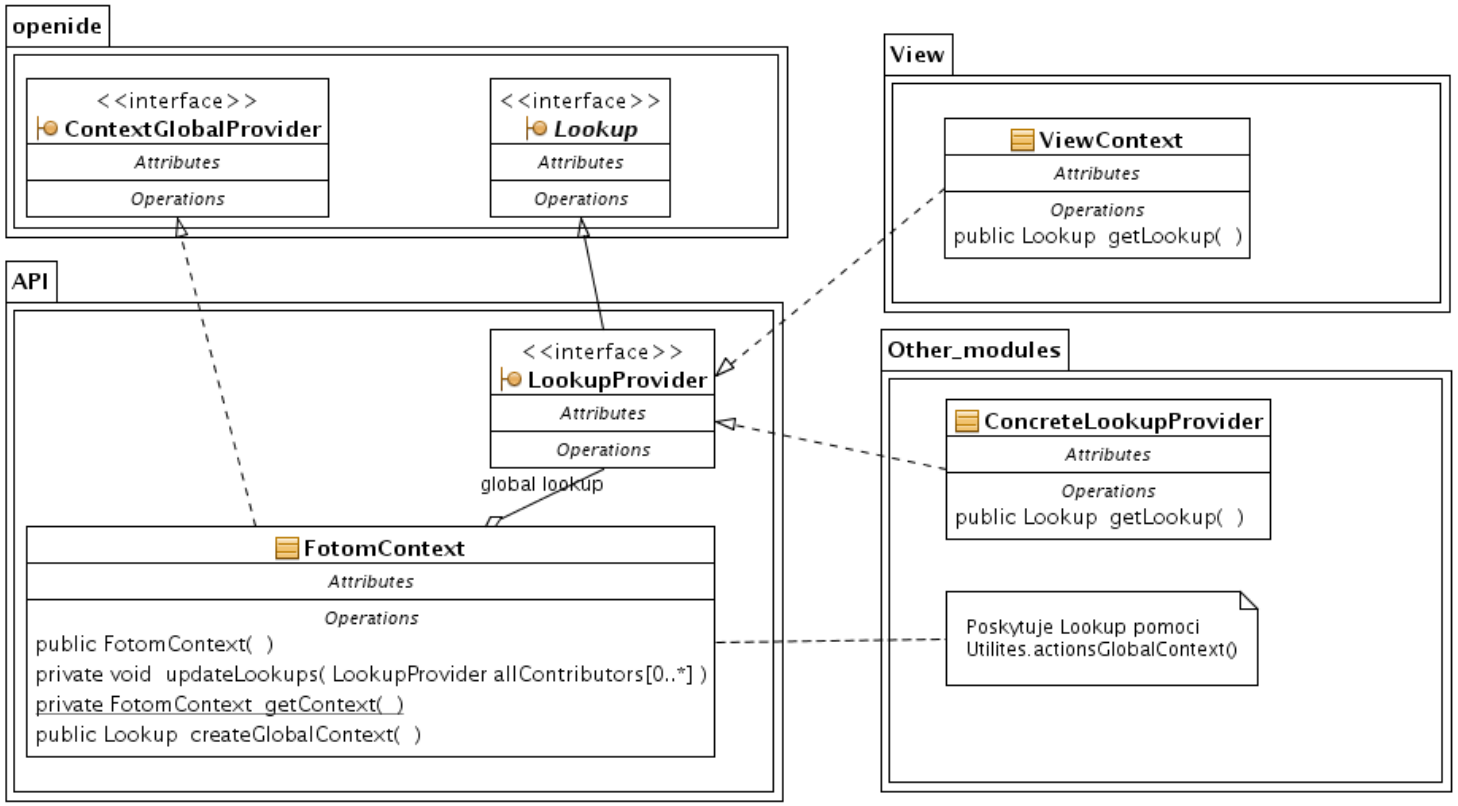
FtmObject Třída *FtmObject* je hlavní třídou nesoucí veškeré informace o snímku a jeho měření. Tato třída spojuje veškeré možné budoucí moduly a poskytuje jim jednotný mechanismus přístupu ke snímku a jeho vlastnostem. Poskytuje mechanismus pro registraci posluchačů, a proto jakýkoli modul může reagovat na jakoukoli změnu týkající se snímku (aplikace filtru atd.). Vedle snímku je hlavním nositelem informace o měření tzv. *kontejner*, který obsahuje veškeré definované objekty, případně lícovací body. Registrací posluchače na FtmObject se tedy docílí i poslouchání změn v nástrojích pro definici oblastí, které jsou na snímku definovány. Je zaručeno, že aktivní snímek, tedy ten, který se právě edituje, je dostupný v kontextovém lookupu (viz. 3.1.3) a dále je zaručeno, že snímek bude v RGB modelu, jelikož při importu snímku ke zpracování se provádí konverze a pokud je původní snímek definován v odstínech šedi nebo RGBA, pak je do RGB transformován. Každý modul se může registrovat k právě aktivnímu snímku poslouchánim lookupu a následným vyhledáním editovaného objektu (viz. výpis 5). Tato vlastnost je dosažena změnou v poskytování *kontextového lookupu* (viz. 4.1.3).

```
FtmObject ftm = Utilities .actionsGlobalContext().lookup(FtmObject.class);
```

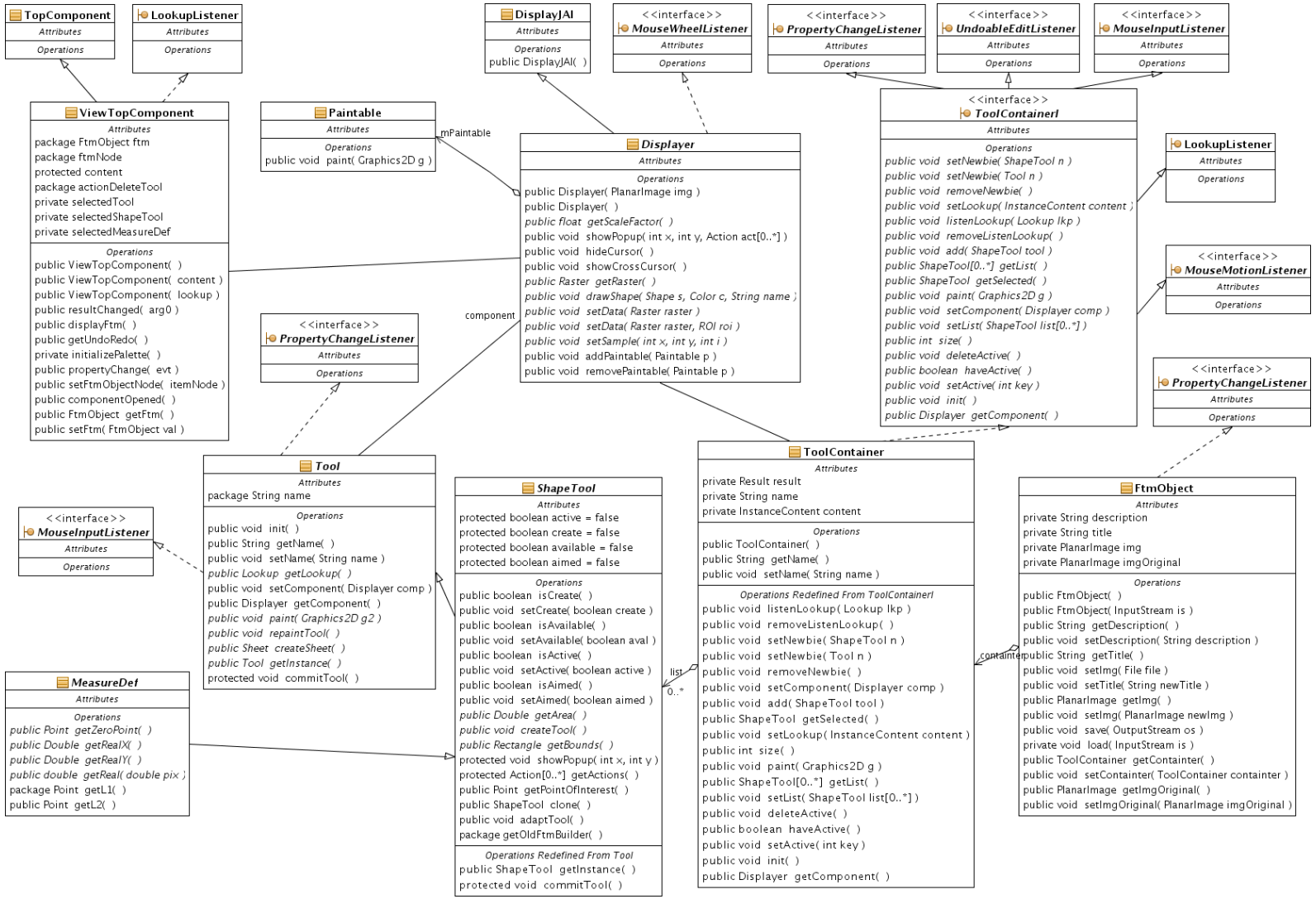
Výpis 5: Získání aktivního snímku pomocí Lookup API

ViewTopComponent Tato třída je odvozena od třídy *TopComponent* (viz. 3.1.4) a její hlavní funkcí je vytvořit okno, do kterého se umístí kreslicí plátno Displyer. Dále poslouchá lookup a pokud se v něm objeví nějaký nástroj (tento nástroj je aktivován), pak je zaobalen do objektu ToolNode, vystaven jako aktivní a *property explorer* tak získá delegované vlastnosti, které může zobrazit. Třída ViewTopComponent si dále vytváří paletu nástrojů, která je definována v layer.xml a poslouchá její události. Pokud dojde k vybrání nového nástroje, pak vytvořený nástroj předává objektu implementující třídu *ToolContainerI*, která zajistí vykreslování nástroje.

Displayer Třída displayer je abstraktní třídou, která určuje jaké metody musí plátno implementovat. Sama implementuje například možnost vkládání objektů typu *Paintable* (viz. dále). Dále přímo poskytuje možnost měnit kurzor a poskytuje metody pro zobrazování nabídky, kterou mohou využít například nástroje zobrazované na plátně. Jelikož



Obrázek 10: Třídní diagram - Kontextový Lookup



Obrázek 11: Třídní diagram - API

se počítá s tím, že budou existovat i nástroje, které budou zasahovat do snímku, displayer poskytuje metody pro získání snímku a možnost zápisu do něj. Obrázek, který třída dává k dispozici, je pouze kopií originálního snímku.

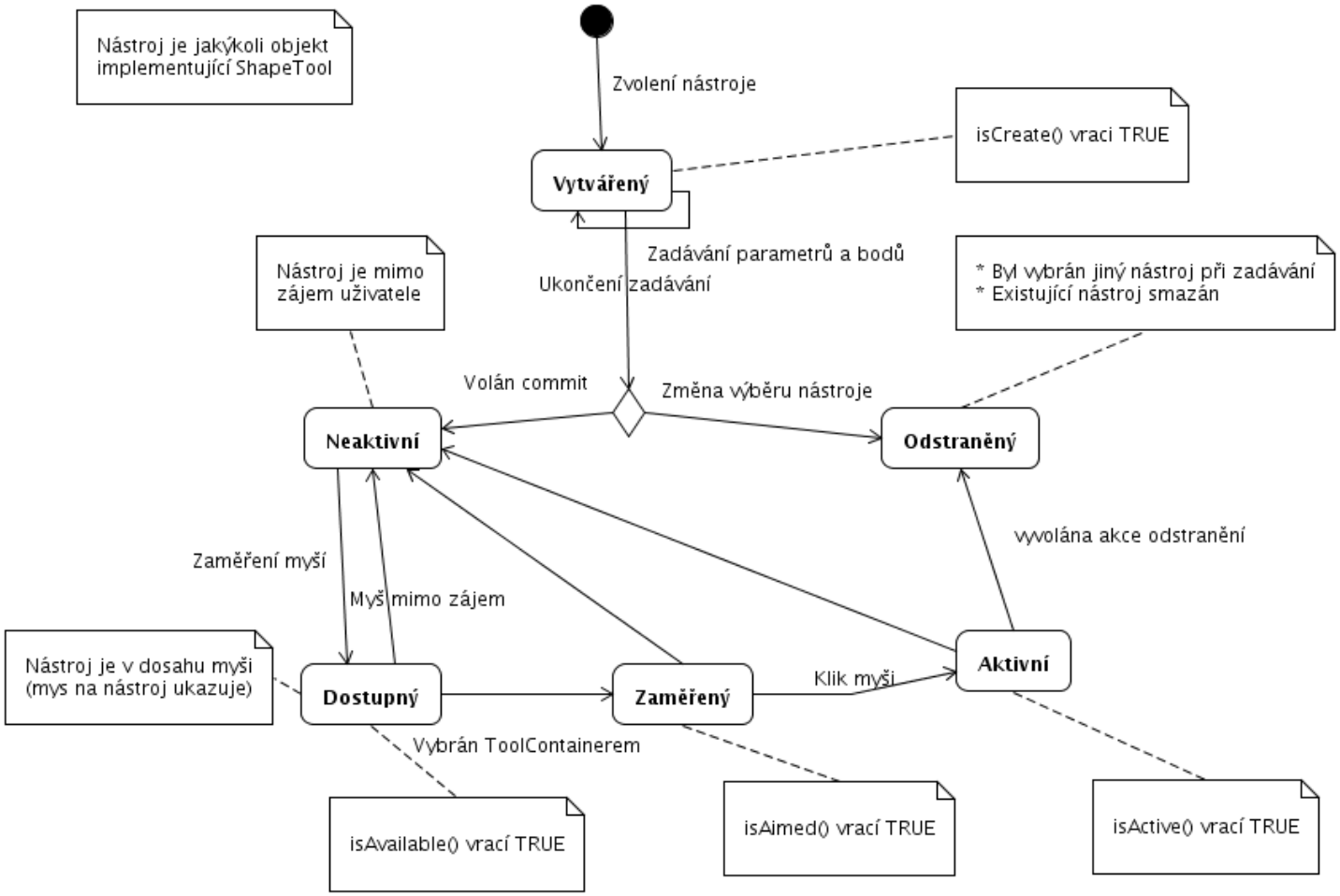
ToolContainerI Toto rozhraní definuje metody, které musí implementovat kontejner. Hlavním úkolem kontejneru je ve svém seznamu uchovávat definované objekty, delegovat události myši a starat se o označení a aktivaci objektů. Implementací tohoto rozhraní tedy vytvoříme prostředníka mezi plátnem a jednotlivými objekty definovanými na snímku, který řídí předávání událostí a poskytuje grafiku plátna a umožňuje tak každému nástroji vykreslit svou reprezentaci.

ToolContainer Třída *ToolContainer* je implementace rozhraní kontejneru ToolContainerI. Z rozhraní vyplývá, že deleguje události myši, ale také podle vstupu myši určuje který z vhodných objektů (instance nástroje třídy ShapeTool) bude vybrán. Dále vkládá aktivní objekt do kontextového lookupu, aby mohl být nalezen ostatními moduly a aby ViewTopComponent mohl delegovat vlastnosti vybraného objektu.

Tool Třída *Tool* základní abstraktní třída pro implementaci nástrojů. Nástroji vycházející z této třídy je umožněno kreslení na plátno (plátno je předáno kontejnerem), vytvořit jiný nástroj a ten nechat vložit do kontejneru a samozřejmě reagovat na události myši. Pokud nástroj chce pracovat přímo s obrázkem a měnit ho, pak si může z kontejneru získat plátno, které nabízí několik možností práce přímo s obrazovými body obrázku. Po ukončení práce nástroje volá *commitTool*, čímž kontejneru a třídě ViewTopComponent sdělí, že práci ukončil. Nástrojům, které jsou odvozeny z tohoto rozhraní, není dovoleno vložit se do kontejneru a slouží proto především ke zpracování samotného obrazu a ne k definici zájmových objektů.

ShapeTool Abstraktní třída *ShapeTool* je odvozena od třídy *Tool* a přidává další možnosti nástroji, který tuto třídu bude implementovat. Základním rozdílem mezi třídami ShapeTool a Tool je, že implementací této třídy umožníme nástroji, aby se i po dokončení úvodní fáze (většinou definice objektu) dále zobrazoval na snímku jako nová vrstva (přímo do snímku nezasahuje). ShapeTool se může nacházet v několika stavech, které z velké míry ovlivňuje on sám. Může být vytvářen, zaměřitelný (říká o sobě, že ho může kontejner vybrat jako zaměřený), zaměřený (určuje kontejner), aktivní a nebo neaktivní. Stavy zaměřitelný a zaměřený se tu objevují, jelikož se může stát, že se objekty překrývají a více objektů může být aktivní, pak tímto stavem sdělí, že na ně ukazuje myš (jsou zaměřitelné) a je už na kontejneru, aby vybral „vítěze“. Znázornění stavů je na obrázku 12.

MeasureDef MeasureDef je poslední ze skupiny abstraktních tříd popisujících chování nástroje. Tato třída je odvozena od *ShapeTool* a přidává další funkčnost potřebnou pro definici referenčních bodů, má za úkol vypočítat poměr mezi skutečnou délkou v definovaných jednotkách, dále obsahuje metody pro přepočet souřadnic z lokálních (na



Obrázek 12: Stavový diagram objektu ShapeTool

snímku) na globální (ve skutečnosti) a umožňuje definovat vlastnosti měření jako jsou jednotky, ve kterých je meření prováděno.

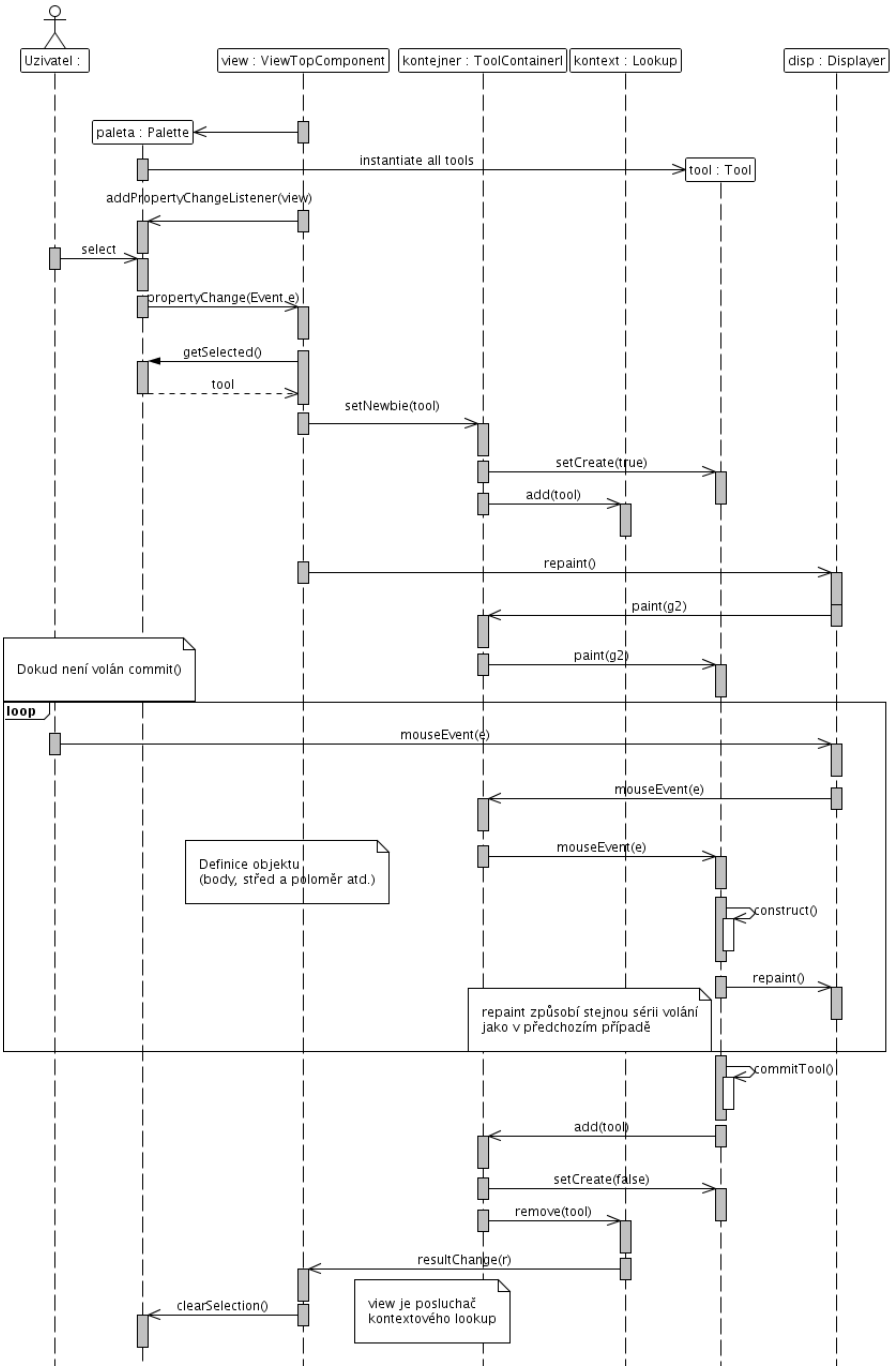
Paintable Rozhraní *Paintable* definuje jednu metodu *paint(Graphics2D d)*, kterou musí třída implementovat, pokud chce cokoli zobrazovat na kreslicím plátně. Tato třída je vhodná pro moduly zobrazující dodatečné informace na plátně (např. výsledky měření) a není určená pro nástroje definující objekty na snímku, kterým prostředky plátna poskytuje kontejner.

Sekvenční diagram Na obrázku 13 je pro lepší popis chování API při výběru nového nástroje a jeho definici zobrazen sekvenční diagram. Při inicializaci třídy ViewTopComponent je vytvořena paleta, definována v layer.xml. Tato paleta má definované jednotlivé nástroje rovněž pomocí XML dokumentů, nástroje do palety může přidávat libovolný modul. Zároveň se třída ViewTopComponent zaregistruje jako posluchač palety. Jakmile uživatel klikne v paletě na nástroj, ViewTopComponent získá zvolený nástroj a předá ho kontejneru ToolContainerI a ten jej vloží do kontextového lookup (ostatní moduly poslouchání kontextového lookup zaznamenají změnu). Zároveň dochází k překreslení kreslicího plátna, které postupně vyzve jednotlivé definované nástroje spolu s novým nástrojem k vykreslení. Následuje cyklus, ve kterém je postupně definován nový objekt (ten je ve stavu „vytvářený“, viz. obrázek 12).

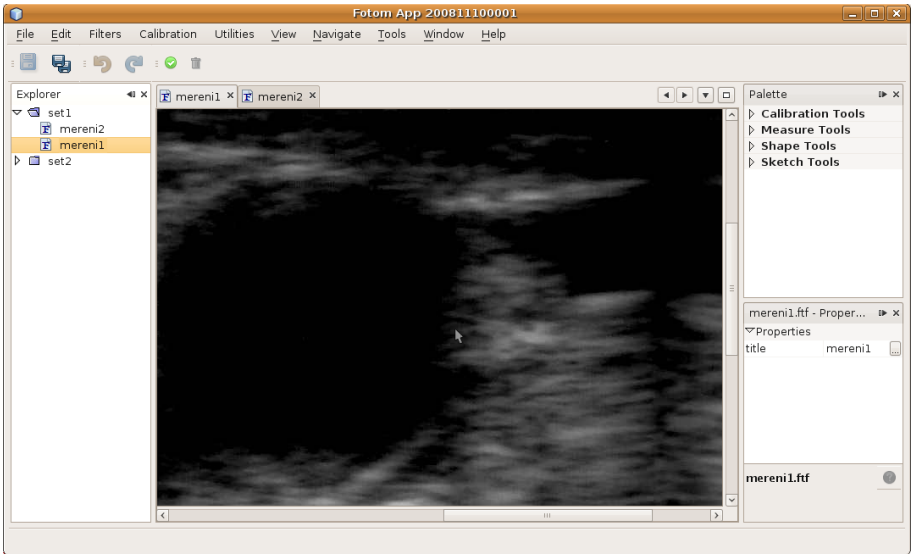
Samotné vytváření objektu není striktně definováno a je pouze znázorněno metodou *construct()*, každá implementace abstraktní třídy *Tool* nebo od ní odvozených tříd si definuje vlastní způsob konstrukce. Třída má pouze předepsané, že musí být schopna reagovat na vstup myši. Jakmile nástroj dokončí svou definici (např. byly definovány všechny body), pak volá předdefinovanou metodu *commitTool()*, která vloží vytvořený a definovaný objekt do kontejneru (dosud byl nástroj pouze v dočasné paměti a kdyby se *commitTool()* nezavolalo, pak instance nástroje zaniká). Tímto je nástroj vytvořen a ve stavu „neaktivní“. Jelikož objekt není ve stavu „aktivní“, je odstraněn z kontextového lookupu.

4.1.4 Implementace

Navržený systém byl implementován na platformě NetBeans (viz. 3.1), z čehož vyplývá, že bylo použito programovacího jazyka Java v1.6. Před začátkem implementace se musel vyřešit problém sdílení a verzování zdrojového kódu, protože návrh a implementace jádra probíhala společně Bc. Janem Králem, který se zaměřil na datovou vrstvu aplikace a jehož práce se dále věnovala kalibraci snímů na vytvořeném jádře. Jako nejvhodnější nástroj byl zvolen Subversion (SVN), což je náhrada za starší systém CVS. Jako vhodné úložiště poskytující SVN a další praktické nástroje pro snadnou spolupráci se ukázal projekt Assembla (http://www.assembla.com/), který v počátcích vývoje zdarma nabízel prostor pro ukládání zdrojových souborů, obrázků a diagramů. Později tento projekt byl zpoplatněn pro komerční využití. Jako klient bylo použito vývojové prostředí NetBeans, které pro tento systém nabízí zásuvný modul.



Obrázek 13: Sekvenční diagram definice nového objektu

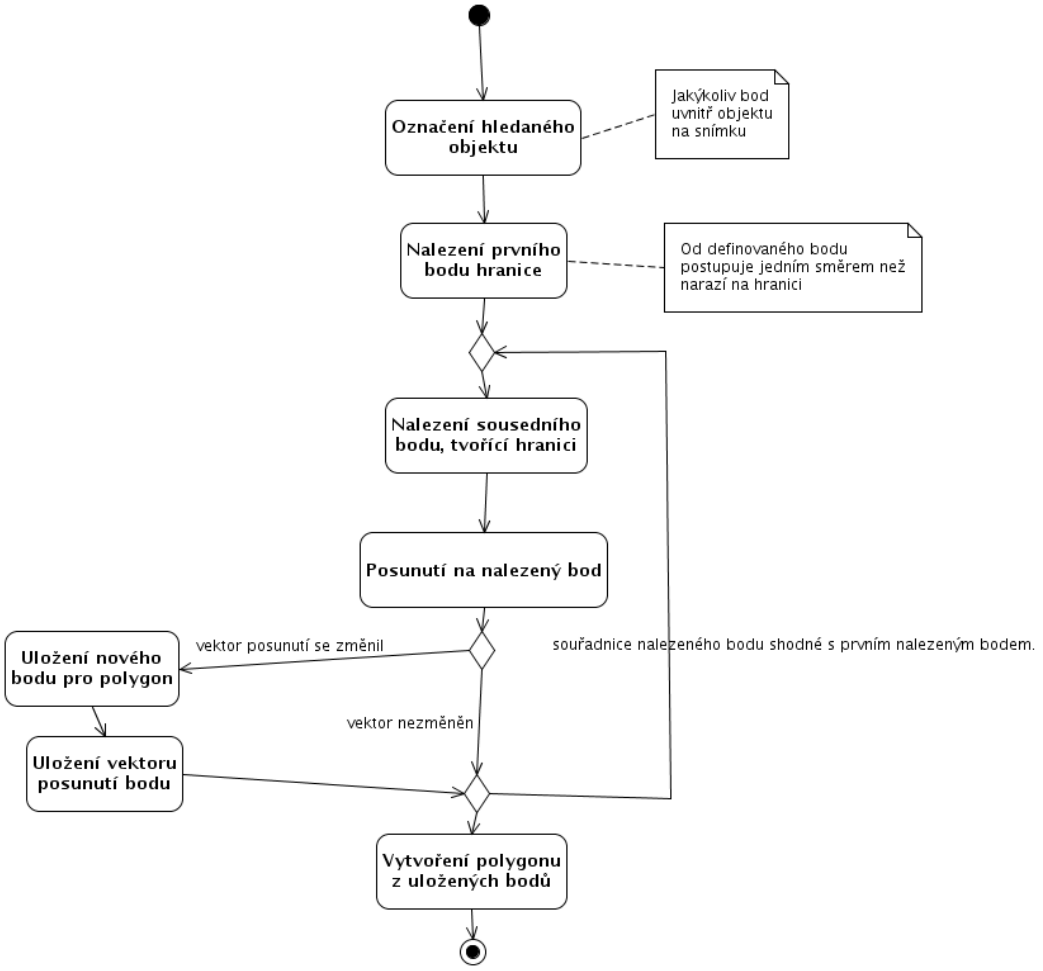


Obrázek 14: Ukázka aplikace - jádro

Pro práci s obrázky bylo zvoleno JAI API (viz. 3.2), které nabízí množství metod pro základní operace s obrázky, ale i pokročilejší metody jako je konvoluce nebo Fourierova transformace. Pomocí JAI je řešeno načítání a export snímků, a proto bude Fotom 2009 schopen zpracovávat téměř libovolný formát obrázku. Třída Display, která implementuje rozhraní Displayer používá JAI pro přibližování a oddalování snímku. Třída display je dále implementována tak, že automaticky přepočítává souřadný systém a upravuje události myši tak, aby žádný implementovaný nástroj se nemusel starat to, zda je snímek přiblížený.

Podle návrhu jsem naimplementoval kontejner ToolContainerI. Jako algoritmus výběru zaměřeného nástroje jsem zvolil metodu, kdy se zjistí obsah objektu (každý nástroj musí implementovat metodu, která vrací obsah) a pokud je zaměřitelných objektů více, pak se zvolí ten, který má nejmenší obsah.

Jádro systému je už spustitelná aplikace, která je schopná vytvořit nové měření a zobrazit snímek na plátně. Dále vytvoří paletu nástrojů, která je prázdná (je na ostatních modulech aby vytvořily konkrétní nástroje). Ukázka aplikace je na obrázku 14.



Obrázek 15: Diagram aktivit hledání hranice objektu

4.2 Moduly pro definici objektů

4.2.1 Byznys modely

Diagram aktivit na obrázku 15 zobrazuje postup při vyhledávání hranice objektu. Nejprve se označí známá část objektu a vyhledá první možná hranice, po které se dále postupuje, dokud se nenarazí na bod, kde hledání začalo.

4.2.2 Specifikace požadavků

Vytvořit modul pro definici objektů na snímku znamená vytvořit sadu nástrojů, které budou schopny pomocí interakce s uživatelem vymezit oblast zájmu na snímku.Umožníme tak další operace se snímky jako je vytvoření 3D modelu, transformace a kalibrace snímku,

ale především měření vzdáleností. Jelikož Fotom 2009 je v prvé řadě program ulehčující měření na snímcích, je nutné se zaměřit na definování oblastí a měření mezi nimi velmi důkladně.

Při měření se velmi často vychází z měření předcházejícího na podobném snímku (např. snímek stejného problému s odstupem času) a zkoumají se odchylky jednotlivých měření, proto nástroje by měly být schopny importu a adaptace na jiný snímek. Objekty jsou definovány téměř výhradně na hranici mezi popředím (světelná stopa v šachtě, nález na ultrazvukovém snímku) a pozadím (především černá barva). Adaptací se rozumí úprava pozice a velikosti objektu podle nového snímku, na který je importován, tedy vyhledání nové hranice, přičemž se počítá s tím, že rozdíly mezi snímky nebudou markantní a nová hranice se bude vyhledávat v malém okolí hranice původní. Pokud se nová hranice nenalezne, pak se objekt (nebo jeho část) neadaptuje, ve velkých vzdálenostech se hranice nevyhledává, jelikož nenalezení nového umístění může být způsobeno zhoršenou kvalitou snímku. API aplikace (viz. 4.1) dává možnost adaptace každého nástroje, ale z hlediska použitelnosti je vhodné tuto funkčnost implementovat pouze u těch, které definují oblast. Ostatní nástroje se budou pouze importovat.
Nástroje pro definování oblastí lze rozdělit na 3 typy:

- Nástroje pro definici referenčních bodů
- Nástroje pro definici oblastí uživatelem (manuální vytvoření objektu)
- Nástroje pro automatické (poloautomatické) definování oblastí
- Nástroje definující mřížku

Definice lícovacích bodů Měření vzdáleností mezi objekty na snímku je možné pouze v obrazových bodech (px), tato jednotka je ale pouze relativní a použitelná pouze pro srovnávání vzdáleností a nic neříká o reálné vzdálenosti. Aby bylo možné měřit i v reálných hodnotách, je nutné na snímku určit lícovací body. Tyto body mají 2 důležité funkce. Definují měřítko mezi vzdáleností v obrazových bodech a reálnou vzdáleností. Nástroj tedy musí umožnit na snímku zadat 2 body a definovat reálnou vzdálenost ve zvolených jednotkách. Druhou funkcí lícovacích bodů bude určení polohy v *globálních* souřadnicích, což je nutné při zpracování série snímků, které zachycují sledovaný problém ve více snímcích, úhlů a nebo v časovém odstupu. Pokud tedy budou definovány lícovací body, pak změřené vzdálenosti se budou přepočítávat podle určeného poměru a *lokální souřadnice* (souřadnice na snímku v obrazových bodech) se budou moci přepočítávat na reálné (globální) umístění objektu. Na každém snímku půjde umístit pouze jeden nástroj definující lícovací body.

Definice objektů uživatelem Definováním objektu na snímku určíme zájmovou oblast (nebo bod), který je možný dále zkoumat. Definice bude umožněna manuálně, tedy tak, že uživatel podle svého uvážení ohraničí sledované oblasti. Pro ulehčení práce je potřeba vytvořit sadu nástrojů, popsaných níže, které budou definovat různé druhy oblastí. Jelikož se počítá s tím, že na snímku může být více takto definovaných objektů, které

mohou do sebe zasahovat, je potřeba umožnit uživateli definovat barvu hranice zobrazeného objektu, barvu výplně a způsob zobrazení (některé objekty nemusí být viditelné pořád), dále u objektů definujících oblast je vhodné zobrazovat i obsah a další zájmové body (střed, těžiště), jelikož k těmto bodům se provádí následné měření.

Nástroj kružnice Pro měření odchylek a vzdáleností na snímcích nejrůznějších šachet kruhového tvaru je vhodné použít kružnici. Kružnice je jednoznačně definována buď svým středem a poloměrem a nebo trojicí bodů, ležících na kružnici. Pro nástroj *kružnice* je vhodnější umožnit uživateli definovat střed a poloměr (ostatně i druhý způsob bude možný viz. dále). Definovaná kružnice musí být dále editovatelná, uživatel musí mít možnost změnit jak poloměr, tak i umístění kružnice (střed). Pro některé druhy měření je vhodnější používat místo kužnice polygon, který je vepsán do kružnice, proto nástroj *kružnice* musí být schopen vytvořit ze zadaného počtu bodu objekt polygon vepsaný do definované kružnice.

Adaptace kružnice by měla probíhat tak, že kružnice se rozdělí na několik bodů, ležících na ní a pro každý bod se vyhledá nové umístění. Vznikne polygon, ze kterého se průměrováním vypočítá kružnice.

Nástroj polygon Velké množství sledovaných oblastí na snímku nemají pravidelný tvar a proto není nutné vytvářet nástroje definující obdélníky či jiné pravidelné útvary s vyjímkou kružnice, která své uplatnění má. Mnohem vhodnější je vytvořit univerzálnější nástroj *polygon*. Polygon se definuje postupným zadáváním bodů. Měření vzdálenosti od objektu polygon se provádí k jeho těžisti, proto nástroj musí být schopen vypočítat své těžiště a zobrazit ho. Samotné zobrazení objektu polygon spočívá ve vykreslení lomené, uzavřené čáry a jednotlivých zadaných bodů. Po definici musí být opět umožněno objekt editovat změnou polohy jednotlivých bodů a posunutím polygonu jako celku. Nástroje, které automaticky definují objekty na snímku, budou nejspíše využívat právě nástroj polygon a v některých případech měření je vhodnější vytvářet kružnici, proto je potřeba umožnit průměrováním vytvořit z polygonu kružnici. Tímto je i umožněna definice kružnice zadáním 3 bodů, které na ní leží (definováním trojúhelníku a následné vypočítání opsané kružnice). Nástroj polygon bude v budoucnu sloužit k definování bodů pro 3D modelování, kde potřebujeme, aby sledovaný objekt na každém snímku ze série měl stejný počet bodů. Nástroje, které budou polygon používat pro definici nalezených objektů, musí mít prostředky jak redukovat (nebo zvýšit) počet vytvořených bodů. Redukcí počtu bodů polygonu sice snížíme přesnost měření, ale umožníme snadnější mapování nalezených bodů v 3D zobrazení.

Adaptace polygonu bude spočívat v postupné adaptaci jednotlivých bodů polygonu. Pro každý bod se vytvoří přímka mezi těžištěm a adaptovaným bodem a na ní se do (rozumě velké) vzdálenosti bude vyhledávat nová pozice hranice a tedy nové umístění bodu.

Nástroj bod Nástroj bod bude sloužit pro označení místa zájmu, pokud toto místo zájmu je přímo viditelné a není potřeba nijak počítat jeho pozici (která je například přímo vyznačena světelnou stopou).

Nástroj průsečík Nástroj průsečík podobně jako *bod* slouží k definici bodového místa zájmu. Na rozdíl od nástroje *bod* je průsečík zadán čtveřicí bodů. Jednotlivé dvojice definují přímku a zájmový bod leží na průsečíku těchto přímek. Tento nástroj je vhodný pro definici bodu v místech, kde na snímku není patrné jeho umístění (zhoršená kvalita snímku), ale je možno ho vypočítat.

Automatická definice objektů Při měření lékařských snímků se velmi často zajímáme o objekty, které jsou velmi členité a jsou obtížně definovatelné ručně, proto je vhodné vytvořit nástroj, který poskytne možnost definovat takovéto objekty automaticky nebo alespoň poloautomaticky (označením části objektu uživatelem). Kvalita snímku nám většinou nedovoluje definici objektu přímo, ale je nutné použít jeden nebo sérii filtrů, které snímek upraví (prahování, rozmazání) a dále upravíme samotný snímek skicováním (dokreslení hranic objektů). Tyto nástroje jsou dále popsány v části 4.3.

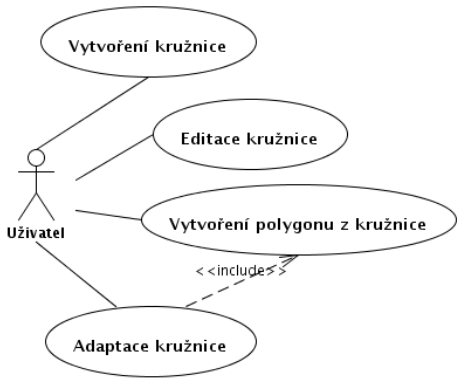
Nástroj tedy bude mít za úkol definovat zvolený objekt tak, že automaticky vyhledá hranici mezi uživatelem zvoleným objektem a pozadím. Po nalezení této hranice nástroj vytvoří *objekt polygon* z předem zvoleného počtu bodů (sám nabídne jejich optimální počet).

Nástroje definující mřížku Pro snadnější definici objektů, ale i pro odměřování vzdálenosti může sloužit nástroj mřížka. Vzhledem k různým typům snímků, pro které se aplikace vytváří, je potřeba vytvořit více druhů mřížky. Pro snímky zhotovené ultrazvukem se vytvoří mřížka ve tvaru paprsků vycházejících z definovaného počátku (sonda ultrazvuku) a kružnice definující vzdálenosti od tohoto počátku. Pro ostatní snímky se vytvoří standardní mřížka (horizontální a vertikální přímky). Samozřejmostí je možnost editace parametrů (vzdálenosti přímek, kružnic).

4.2.3 Manažer objektů

Na snímku se může a často vyskytovat celá řada definovaných objektů, a pokud jich bude velké množství, pak může být snímek nepřehledný. Z tohoto důvodu je potřeba vytvořit panel s manažerem objektů. Tento modul bude zobrazovat seznam jmen definovaných objektů na snímku ve zvláštním okně, které se umístí do levé části aplikace. Manažer bude reagovat na výběr objektu na plátně tak, že v seznamu zvýrazní název objektu který je právě aktivní a nebo pokud uživatel označí jméno v seznamu, pak se tento objekt aktivuje i ve snímku.Tímto způsobem zajistíme větší přehled a bude více jasné, se kterým objektem pracujeme.

Další funkcí manažera bude proměřování a vizualizace vzdáleností mezi objekty. Umožní tedy vybrat dva a více objektů v seznamu a na plátně zvolené objekty spojí



Obrázek 16: Diagram aktivit nástroje kružnice

úsečkou a zobrazí vzdálenost mezi nimi. Pokud budou definovány lícovací body, pak se bude uvádět vzdálenost reálná, jinak pouze v obrazových bodech.

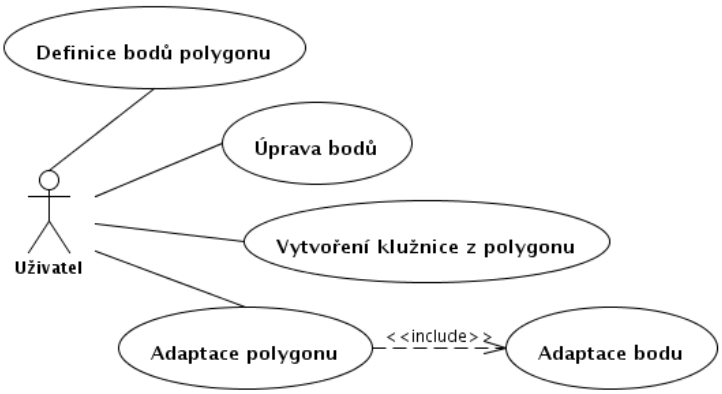
Diagramy případů užití V následujících diagramech (obrázky 16, 17, 18) jsou zobrazeny případy použití vybraných, netriviálních nástrojů pro definici objektů.

Sekvenční diagramy Sekvenční diagram na obrázku 19 popisuje export definovaných objektů (objekty na snímku jsou definovány pomocí nástrojů polygon, kružnice atd.) do souboru programu Fotom2008. Nejprve **director** exportu (objekt řídící export) získá z aktuálního snímku kontejner nástrojů a každý definovaný nástroj požádá o vytvoření třídy umožňující export. Pokud nástroj neumožňuje export (nástroj, který není ve verzi programu 2008 implementován), pak nevrací nic (resp. null). Od každého vytvořeného konkrétního sestavovatele *OldFtmBuilder* director získá potřebné údaje pro export, jako je seznam bodů, jméno, vlastnosti, a vytvoří soubor s popisem objektů, ke kterému exportuje **originální** snímek.

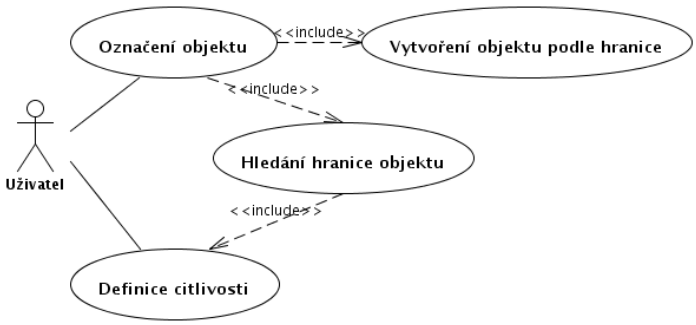
4.2.4 Analýza a návrh

Z definice požadavků vyplývá, že je třeba vytvořit sadů nástrojů (polygon, kružnice, bod, mřížka, průsečík), které budou implementovat abstraktní třídu *ShapeTool* (viz. API 4.1.3), tato třída umožňuje definici objektu a jeho zobrazení na snímku a podporuje měření. Nástroj pro automatickou definici objektu slouží k jednorázové detekci hranice a jeho úkolem je vytvořit objekt polygon, proto je vhodnější implementovat třídu *Tool*.

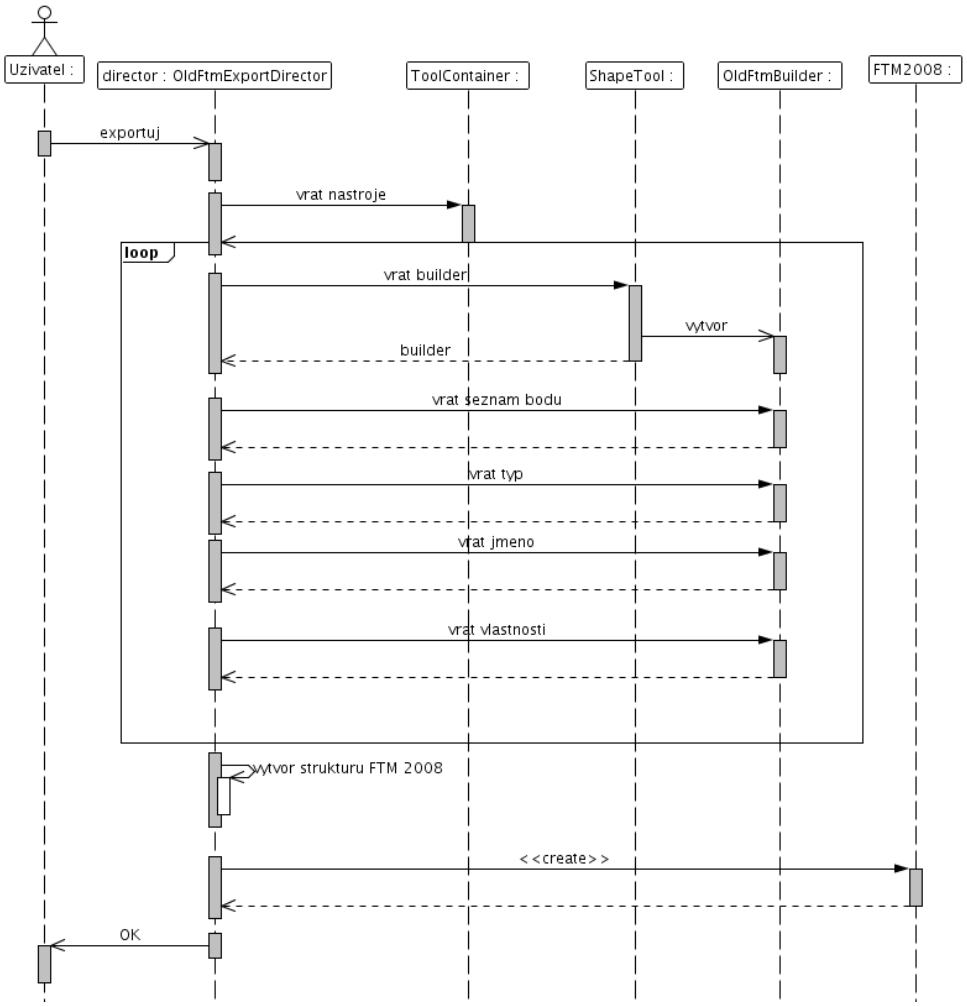
Pro řešení exportu do souboru typu FTM2008 se nabízí použít obdobu návrhového vzoru **Builder** (znázorněn na obrázku 20). Aby se umožnilo exportovat i ostatním budoucím nástrojům v jiných modulech, bude třída implementující director a abstraktní třída builder přesunuta do API.



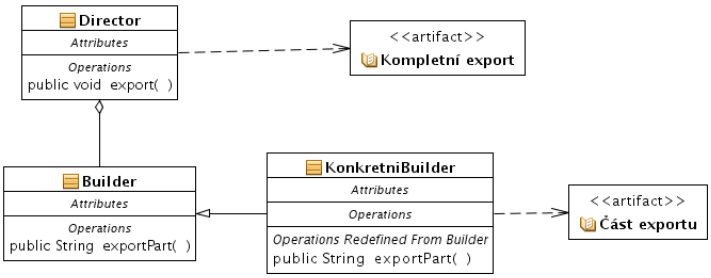
Obrázek 17: Diagram aktivit nástroje polygon



Obrázek 18: Diagram aktivit automatického vyběru



Obrázek 19: Sekvenční diagram exportu do FOTOM2008



Obrázek 20: Návrhový vzor Builder

Většina nástrojů má velké množství vlastností (barva, velikost, styl zobrazeni), které vyplynuly z požadavků na přehlednost a použitelnost. Je vhodné, aby si každý uživatel mohl nastavit preferované vlastnosti a nebyl nucen každému vytvořenému objektu nastavovat svou barvu atd. Platforma Netbeans nabízí možnost vytvoření panelu *nastavení* pro každý nástroj (viz. 3.1.8).

Integrace vytvořených nástrojů do aplikace spočívá ve vložení tlačítka do palety nástrojů, kterou vytváří ViewTopComponent (viz. 4.1.3). V souboru *layer.xml* vytvoříme definici pro vkládání nových nástrojů (výpis 6), kde „ToolPalette“ určuje paletu, kterou hledá ViewTopComponent, „Shape Tools“ definuje novou kategorii v paletě a odkaz na soubor „Nastroj.xml“ určuje samotné vložení nástroje.

```
<folder name="ToolPalette">
        <folder name="Shape Tools">
            <file  name="Nastroj.xml" url="cesta/Nastroj.xml" />
        </folder>
    </folder>
</filesystem>
```

Výpis 6: Integrace nástroje do palety

Definice samotného nástroje znamená pouze určit třídu, která nástroj implementuje (vždy musí být implementací ShapeTool, MeasureDef nebo Tool), definice ikon a pojmenování (viz. výpis 7)

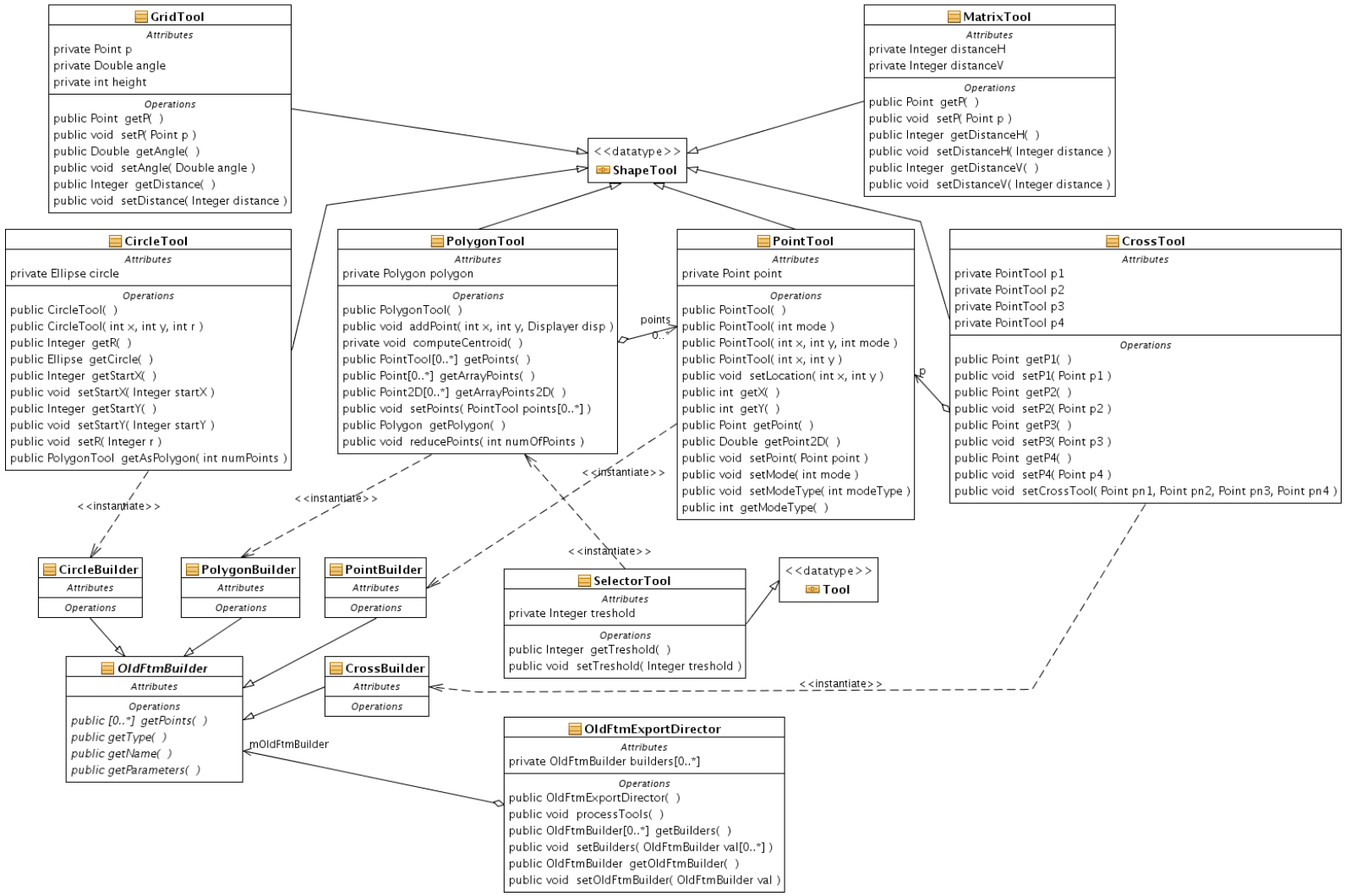
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.1//EN"
  "http :// www.netbeans.org/dtds/editor-palette-item-1_1.dtd">
<editor_palette_item  version="1.0">
    <class name="cesta.k.nastroji.TridaNastroje" />
    <icon16 urlvalue="cesta/k/nastroji/ico/Nastroj16.png" />
    <icon32 urlvalue="cesta/k/nastroji/ico/Nastroj.png" />
    <inline-description>
        <display-name>Nastroj</display-name>
        <tooltip> Nastroj  definujici  ...  </tooltip>
    </inline-description>
</editor_palette_item>
```

Výpis 7: Definice nástroje v XML

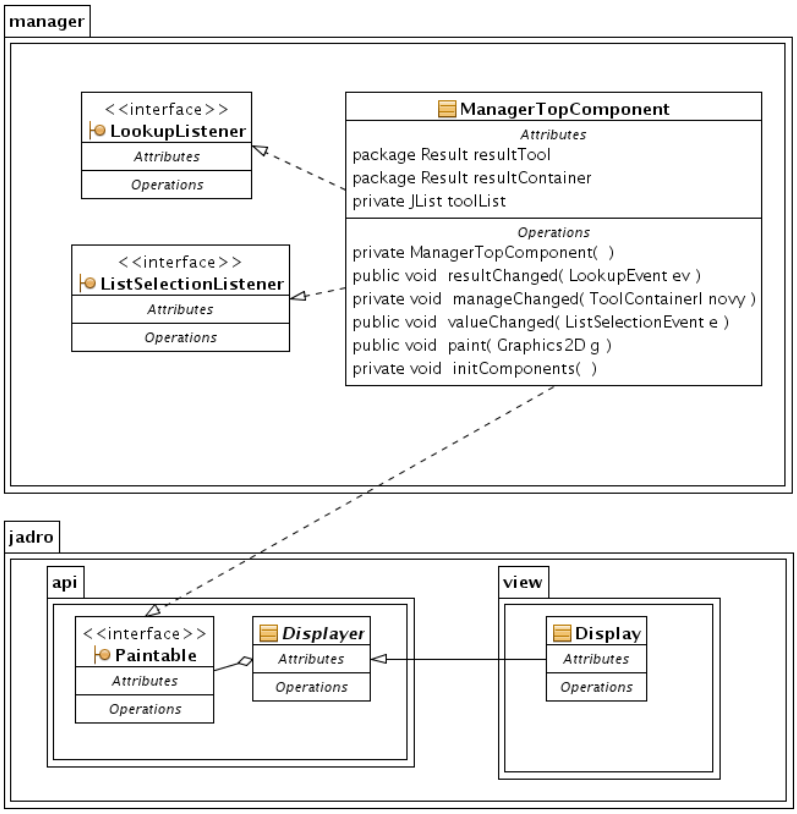
Diagram tříd Diagram tříd na obrázku 21 zobrazuje statický pohled na nástroje definující objekty. Třídy *OldFtmBuilder* a *OldFtmExportDirector* fyzicky patří do modulu API, ale spolupracují pouze s nástroji popsanými zde, proto jejich statický popis je znázorněn až zde. Na obrázku 22 je zobrazen nový modul manažera objektů. Manažer má přístup ke kontejneru a může aktivovat nebo deaktivovat objekty, dále implementuje rozhraní *Paintable* a proto může na plátně zobrazovat výsledky měření.

4.2.5 Implementace

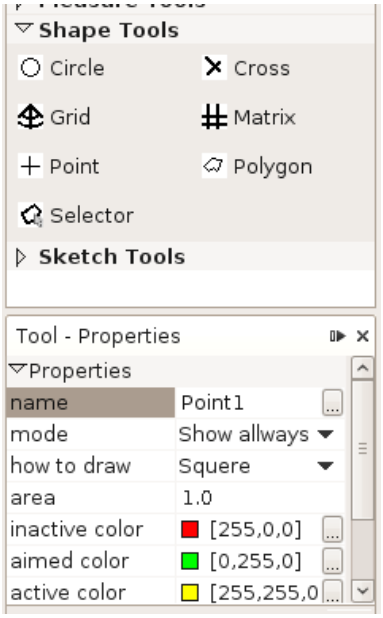
Průvodcem Netbeans se vytvořil nový modul s názvem *Basic Poins*, který je závislý na modulu jádra fotom-api (a dalších modulech Netbeans Platform). Při implementaci jed-



Obrázek 21: Třídní diagram nástrojů



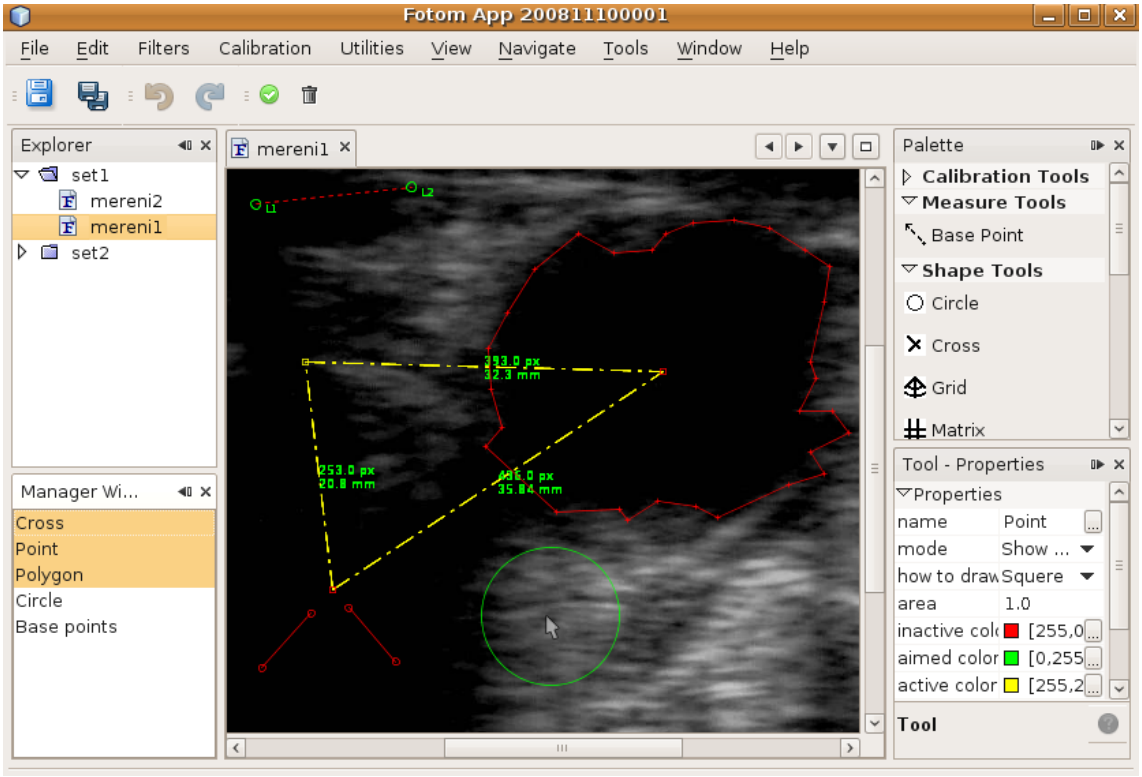
Obrázek 22: Třídní diagram manažera nástrojů



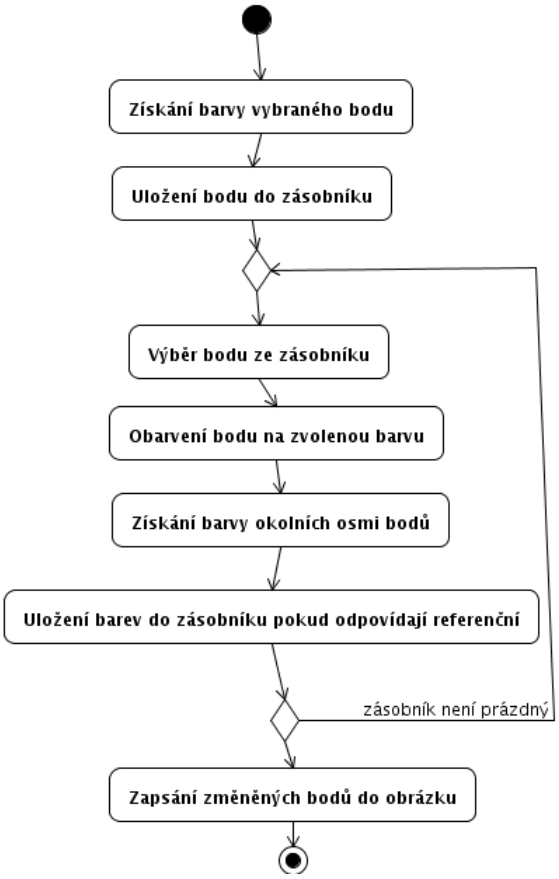
Obrázek 23: Ukázka modulu nástrojů pro definici objektů

notlivých nástrojů byly použity abstraktní třídy ShapeTool, Tool a MeasureDef z API, jelikož jedině implementací těchto tříd umožníme integraci nástrojů do palety. Takto byly vytvořeny třídy CircleTool, PolygonTool, PointTool, CrossTool a ke každé z nich byl vytvořen konkrétní builder schopný exportu. Vytvořené nástroje zachovávají podmínky pro perzistenci 4.1.3 a poskytují přístupové metody pro proměnné, které mají být uloženy. Pro editaci základního nastavení každý nástroj vytváří objekt *Sheet* a vytvoří svou množinu editovatelných vlastností. Třída SelectorTool, která je implementací nástroje pro poloautomatickou detekci objektu na snímku, využívá prostředky JAI (viz. 3.2) pro získání jednotlivých pixelů. Na obrázku 23 je zobrazena ukázka palety a nastavení nástroje. Způsob používání nástrojů je popsáno v *uživatelské příručce*.

Implementace manažera objektů spočívala ve vytvoření nového modulu a vytvoření třídy ManagerTopComponent, která vychází z třídy TopComponent. Pro vytvořenou komponentu se mi podařilo definovat pozici umístění okna do levého dolního rohu, která není standardně podporována. O veškeré panely se stará okenní manažer platformy, a proto si uživatel může bez sebemenších problémů okno manažera přetáhnout kamkoli. Ukázka použití manažera je na obrázku 24, jsou zde zobrazeny i definované objekty polygon, bod, průsečík a kružnice a pomocí manažera se měří vzdálenost mezi třemi vybranými.



Obrázek 24: Ukázka modulu manažer



Obrázek 25: Diagram aktivit vyplňování barvou

4.3 Moduly pro skicování a filtry

4.3.1 Byznys modely

Na obrázku 25 je zobrazen aktivitní diagram vyplňování obrázku barvou.

4.3.2 Skicování - specifikace požadavků

Pokud definujeme objekty na snímku pomocí nástrojů, které jsou schopny detekovat hranici mezi popředím a pozadím, nastává často problém, že vzhledem ke kvalitě snímku bývá úspěšnost správné detekce malá. Z tohoto důvodu potřebujeme nástroje, které nám pomohou upravit snímek tak, abychom jasně oddělili popředí, tedy zájmové objekty na snímku a pozadí. Tyto nástroje můžeme rozdělit na 2 kategorie:

Skicovací nástroje jsou takové nástroje, kterými budeme schopni upravit nedostatky snímku tak, že dokreslíme nebo zvýrazníme přerušené a nejasné hranice. Skicovací nástroje přímo mění obrázek (ve skutečnosti mění jeho kopii, protože plátno

nedovoluje originální snímek editovat) a na rozdíl od nástrojů, které definují objekty, nejsou skicovací nástroje ukládány do kontejneru a úpravou obrázku jejich instance zaniká.

Fitry jsou nástroje, které upravují snímek jako celek buď v prostorové nebo frekvenční doméně. Ve prostorové doméně filtrování nazýváme operaci, která každý bod obrazu průměruje okolními body. Samotný filtr je definován maticí, která nastavením svých koeficientů definuje váhu každému bodu v okolí zpracovávaného bodu. Pro zpracování snímku ve frekvenční doméně se používá Fourierova transformace, která obraz převede do frekvenční oblasti, ve které lze snímek filtrovat pomocí vhodných funkcí [4]. Pro potřeby tohoto modulu postačí filtrování v prostorové doméně.

Úsečky a lomené čáry Tyto nástroje budou schopny kreslit do snímku zvolenou barvou úsečky, které budou mít definovanou tloušťku. Je vhodné při definici úsečky nejprve zobrazovat její náhled, tedy jak bude snímek upraven a po potvrzení zadání úpravu provést. Lomená čára je definována sérií na sebe navazujících úseček a křivek.

Křivky Stejně jako úsečky budou mít i křivky definovatelnou barvu a tloušťku čáry. Sledované objekty na snímku nemají většinou charakter geometrických útvarů, a proto pro dokreslení částí, které jsou zachyceny špatně a nebo úplně chybí, je vhodnější používat křivky. Vytvoříme nástroje pro definici kvadratické Bézierovy křivky, která je zadána třemi body a kubické Bézierovy křivky, která je zadána čtyřmi body [4]. Podobně jako u úseček požadujeme nejprve náhled vytvořené křivky, která se po potvrzení zapíše do snímku.

Zvýraznění hranice a plechovka barvy Nástroj plechovka barvy je dobře známá a nachází se snad ve všech aplikacích, které upravují obrázky. Úkolem plechovky je vyhledat a vyplnit spojitou oblast zvolenou barvou. Úkol nástroje, pro zvýraznění hranice je obdobný jako vyplňování, ale barva se zapíše pouze na okraj nalezeného spojitého objektu.

Filtry Pro potřeby zpracování snímku za účelem zvýšení kontrastu mezi objekty a pozadím je vhodné implementovat následující základní filtry.

prahování: Tato funkce je základním nástrojem pro posílení kontrastu mezi objekty a pozadím. Vstupem funkce je práh (svítivost 0-255) a pokud hodnota svítivosti zpracovávaného bodu (některá z jeho tří složek) je menší než stanovený práh, pak se nastaví na předem definovanou hodnotu (většinou 0). Pokud naopak je hodnota větší než stanovený práh, pak se obvykle bod nastaví na hodnotu 255, případně se neděje nic. Pro prahování je vhodné barevný snímek převést na odstíny šedi, jinak se musí zpracovávat každá z jeho složek (RGB) zvlášť.

detekce hran: Tento nástroj vhodným nastavením konvoluční matice (filtru) umožňuje nalezení oblastí pixelů, ve kterých se podstatně mění jas.

rozmazání: Poslední důležitý nástroj je rozmazání. Podobně jako v nástroji detekující hrany se vhodným nastavením matice filtru způsobí rozmazání, které je vhodné použít k odstranění šumu

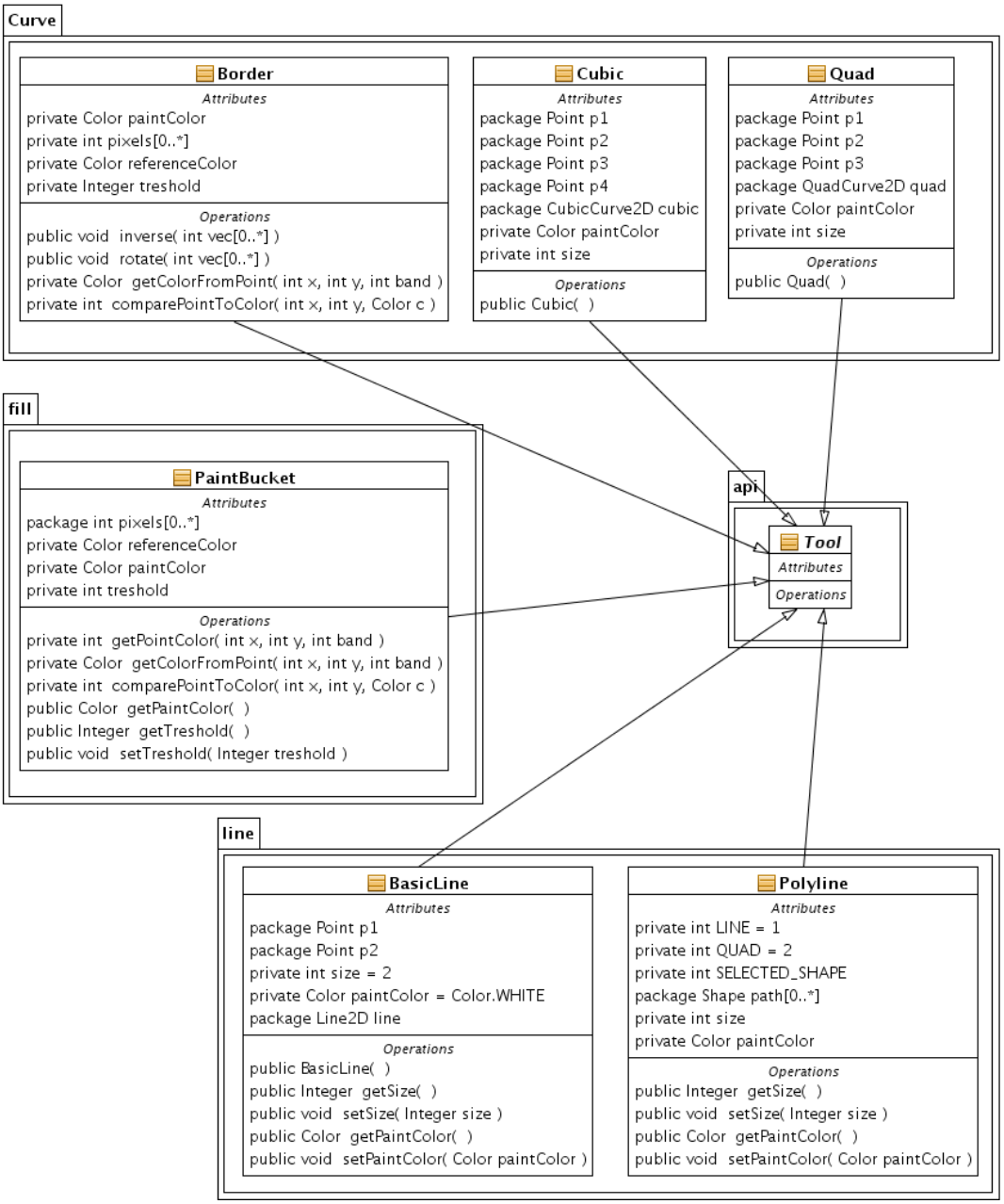
Jako doplňkové nástroje budou vytvořeny filtry pro zaostřování snímku, inverzi barev a převedení barevného obrázku na obrázek ve stupních šedi. Matice použité pro sestavení filtrů budou uvedeny v informativní části uživatelské příručky.

4.3.3 Analýza a návrh

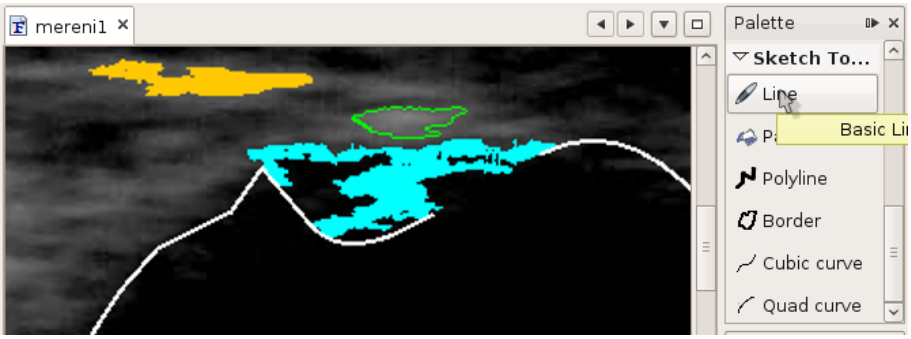
Diagram tříd Na obrázku 26 je zobrazen statický pohled na strukturu skicovacích nástrojů pomocí třídního diagramu. V balíku *Curve* jsou znázorněny třídy *Cubic* a *Quad*, které jsou reprezentací křivek. Do stejného balíku patří i třída *Border*, která reprezentuje nástroj, který zvýrazní hranici objektu. Třída definuje metody pro rotaci a inverzi vektoru, který sleduje směr pohybu hranice, dále má metody pro získání barvy určeného bodu a jeho porovnání. Podobné metody má i třída *PaintBucket*, která implementuje algoritmus vyplňování a reprezentuje nástroj „plechovka barvy“. Poslední skupinou jsou třídy *BasicLine* a *Polyline* z balíku *line*. Tyto třídy reprezentují nástroj schopný kreslit jednoduchou a lomenou čáru. Všechny třídy implementují abstraktní třídu *Tool*, která je vytvořena právě pro nástroje, které chtějí zasahovat přímo do snímku a nepotřebují být vykreslovány jako další vrstva, a proto se nevkládají do kontejneru objektů.

4.3.4 Implementace

Byl vytvořen nový modul **fotom-sketch-tools** se závislostí na **fotom-api** a každý nový nástroj byl zařazen do palety nástrojů deklarací v souboru layer.xml. Implementace nástroje plechovky barvy spočívala v implementace semínkového algoritmu vyplňování, tento algoritmus je původně navržen jako rekurzivní, ale ukázalo se, že pro velké plochy není vhodný, protože docházelo k přetečení zásobníku, a proto byl upraven do lineární podoby. Byl přidán parametr *citlivost*, který určuje rozsah jak moc se může barva změnit, aby se obrazový bod považoval jako součást objektu, protože se barva nevyplňila tak, jak by uživatel pohledem očekával. Nástroj který slouží jako označení hranice se nejprve zdál velmi podobný problému vyplňování barvou, nakonec byl ale zvolen algoritmus používaný při automatické detekci objektu, jelikož je s tímto nástrojem velmi podobný jen s rozdílem, že po detekci hranice se nevytváří polygon na snímku jako nová vrstva, ale mění se přímo body snímku na zvolenou barvu. Obě křivky byly naprogramovány za použití standardních tříd jazyka Java (*CubicCurve2D*, *QuadCurve2D*). Při programování lomené čáry se nabízela možnost použití třídy *GeneralPath*, která umožňuje vytváření lomené čáry, ale nakonec byl zvolen způsob vytváření základních tvarů (úsečka, křivka), kdy se každá vytvořená část ihned zapíše do snímku. Ukázka nástrojů aplikovaných na snímku je na obrázku 27.

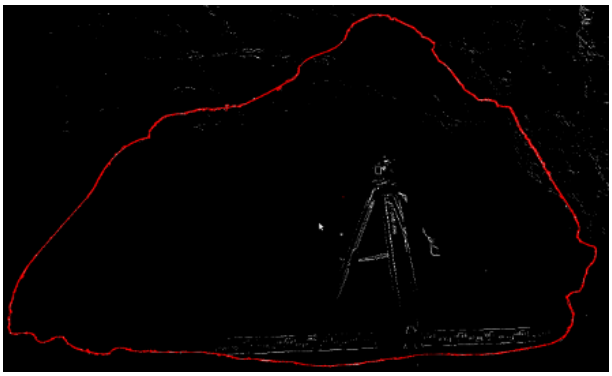


Obrázek 26: Třídní diagram skicovacích nástrojů



Obrázek 27: Ukázka skicovacích nástrojů

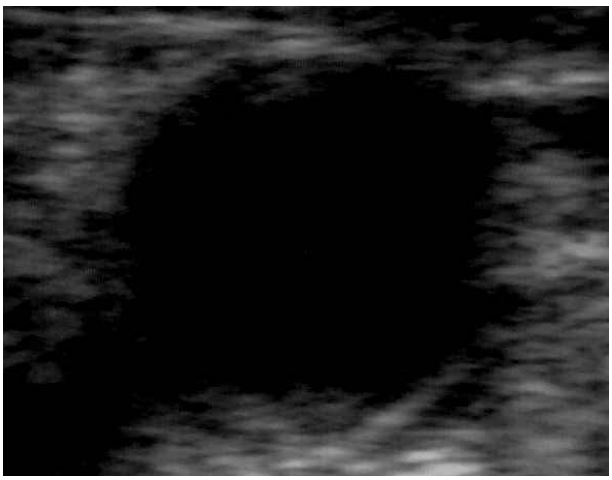
Obrázek 28: Originální snímek Jáchymky



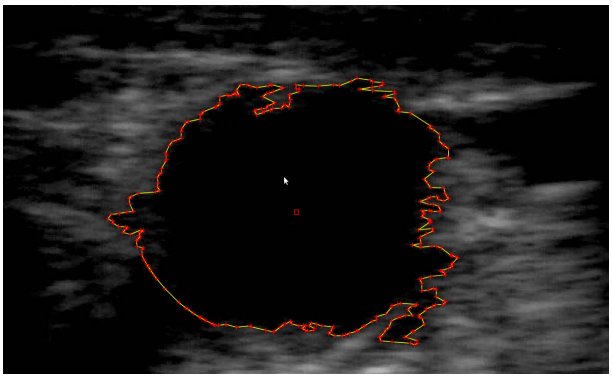
Obrázek 29: Snímek Jáchymky zpracovaný systémem Fotom 2009

4.4 Ověření funkčnosti navrženého systému

Vytvořené jádro systému spolu s implementovanými moduly pro definici objektů, měření a skicování jsem úspěšně ověřil na lékařských snímcích krční tepny, snímcích dolů ČSA OKD, a.s a na snímku jeskyně Jáchymka pořízeném v Moravském krasu studenty VUT Brno. Na následujících obrázcích, které vznikly prahováním a aplikací filtrů jsou zobrazeny nalezené objekty. Obrázek 28 zobrazuje původní snímek jeskyně, na kterém je znázorněna laserem vytvořená světelná stopa. Obrázek 29 zobrazuje zpracovaný snímek s vytvořeným polygonem, který je definován pomocí 6000 bodů. Na obrázku 30 je zobrazen ultrazvukový snímek krční tepny, která je na obrázku 31 označena polygonem. Dále jsem ověřil platformovou nezávislost aplikace úspěšným spuštěním a chodem aplikace na platformách Linux (distribuce Ubuntu 8.10) a Windows (Windows XP, Windows Vista). Na platformě Windows Vista jsem ověřil schopnost systému spolupracovat s programem Fotom 2008 a úspěšně jsem exportoval veškeré podporované objekty.



Obrázek 30: Originální snímek krční tepny



Obrázek 31: Snímek tepny zpracovaný systémem Fotom 2009

5 Závěr

Ve své diplomové práci jsem se seznámil s fotogrammetrií, vědním oborem zabývajícím se zpracování fotografií za účelem měření a vyhodnocování údajů. V úvodu jsem se zaměřil na fotogrammetrický program Fotom 2008, který byl vyvinut na katedře informatiky FEI VŠB TU Ostrava. Seznámil jsem se s jeho možnostmi a současnými požadavky na moderní systém a na jejich základech jsem popsal požadavky a analyzoval nové jádro systému Fotom 2009. Pro stavbu nového systému byla vybrána platforma NetBeans, kterou jsem popsal v části zabývající se analýzou softwarových prostředků. Podle požadavku na snadnou rozšiřitelnost a univerzálnost systému bylo vytvořeno veřejné API, na kterém se implementovalo jádro. Propracované API umožňuje jednoduchý přístup ke zpracovávanému snímku a poskytuje snadný způsob komunikace mezi budoucími moduly. Dále definuje, jak mají vypadat případné nástroje, které budou se snímkem pracovat.

V další části práce jsem analyzoval požadavky na nástroje pro definici zájmových bodů a objektů na snímku a implementoval modul poskytující paletu nástrojů pro jejich definici a modul pro měření. Umožnil jsem manuální definici zájmových objektů a dále jsem vytvořil nástroj, který je schopen označit zájmový objekt automaticky. Každý vytvořený nástroj poskytuje uživateli informace o svém obsahu či velikosti a pomocí svých vlastností může uživatel měnit jeho chování. Každý nástroj je kompatibilní s předchozí verzí programu, jelikož systém Fotom 2008 poskytuje řadu funkcí, které nejsou prozatím v nové verzi implementovány. Aktuální systém je schopen spouštět starší verze a tím je zajištěna plná zpětná kompatibilita a umožněn snadný a plynulý přechod na novou verzi programu. Pro budoucí využití vytvořených nástrojů v modulech, které mohou porovnávat snímky, jsem implementoval možnost importovat definované objekty z již provedeného měření do nového snímku a vytvořil jsem sytém adaptace, který spočívá v automatickém hledání nové pozice zájmového bodu v blízkém okolí původního umístění.

Při měření se můžeme setkat se sníženou kvalitou snímků. Obzvláště na snímcích pořízených pomocí ultrazvuku se potýkáme s šumem a nejasnými hranicemi mezi zájmovým objektem a pozadím. Tyto problémy způsobují obtížně proveditelné měření, a proto jsem vytvořil modul, který obsahuje sadu nástrojů schopných dodatečně snímek editovat a kreslit do něj, pokud potřebujeme doplnit chybějící části objektů. Vzniklé nástroje jsem doplnil sadou filtrů, sloužící pro redukování šumu a prahování snímku, tedy k lepšímu oddělení zájmových objektů od pozadí.

Jádro systému spolu s moduly tvoří komfortní systém, který je navržen pro měření na lékařských snímcích a snímcích z hornictví. Cílem bylo vytvořit nový systém Fotom na základě moderních softwarových prostředků s důrazem na přívětivé uživatelské rozhraní, intuitivní ovládání a jednoduchou rozšiřitelnost, což se podařilo. Funkčnost jádra systému se všemi implementovanými moduly jsem úspěšně ověřil na sérii snímků z hornictví a lékařství. V současné době Bc. Jan Král vytváří unikátní modul, který umožní kalibrovat lékařské snímky pořízené endoskopickou sondou a umožnit tak přesné měření a následnou diagnózu. V budoucnosti bych navrhoval vytvořit moduly pro porovnávání dvou a více měření a umožnit sledovat vývoj sledovaných objektů v čase nebo prostoru. Pro lepší vizualizaci bych vytvořil modul pro 3D zobrazení a animaci sledovaných objektů a tím nahradil celý systém Fotom2008.

Na adrese http://fotomapp.tk jsem vytvořil internetovou prezentaci aplikace, kde je program dostupný ke stažení a kde se i nachází repozitář pro automatické aktualizace. Aplikace je nastavena, aby automaticky aktualizace kontrolovala a nové verze oznamovala a nabízela ke stažení. Součástí vytvořené aplikace jsou zdrojové kódy, uživatelská a programátorská příručka.

6 Literatura

[1] LIČEV Lačezar: *Systém FOTOM 2007 a vizualizace procesu měření* Ostrava:Tanger, spol. s.r.o., 2008, vol. 2008, čis. GIS2008, GIS Ostrava, 49-50, Tanger, spol. s.r.o., ISBN 978-80-254-1340-1

[2] LIČEV Lačezar: *Systém FOTOM 2008 a vizualizace procesu měření* Ed. Ing. Kateřina Pešková, Ostrava:TANGER, spol.s.o., 2009, vol. 2009, čis. 1, 35, VŠB TU Ostrava, HGF, Institut geoinformatiky, ISBN 978-80-87294-00-0

[3] BOUDREAU Tim, TULACH Jaroslav, WIELENGA Geertjan: *Rich Client Programming: Plugging into the NetBeans Platform* 1.vyd. Prentice Hall, 2007. 640 s. ISBN 01-3235-480-2.

[4] ŽÁRA Jiří, BENEŠ Bedřich, SOCHOR Jiří, FELKEL Petr: *Moderní počítačová grafika* 2.vyd. Brno: Computer Press, 2004. 609 s. ISBN 80-251-0454-0.

[5] BöHM, Jozef: *FOTOGRAMMETRIE* [online]. 2002 [citováno 15.dubna 2009] Dostupné na: <http://igdm.vsb.cz/igdm/materialy/Fotogrammetrie.pdf>

[6] Sun Microsystems, Inc: *Programming in Java Advanced Imaging* [online]. 1999 [citováno 10.ledna 2009] Dostupné na: <http://java.sun.com/products/java-media/jai/forDevelopers/jai1_0_1guide-unc/>

[7] Sun Microsystems, Inc.: *Java Advanced Imaging* [online]. 2009 [citováno 18.ledna 2009] Dostupné na: <http://java.sun.com/javase/technologies/desktop/media/jai/>

[8] Sun Microsystems, Inc. : *Modules API* [online]. 2009 [citováno 2.dubna 2009] Dostupné na: <http://bits.netbeans.org/dev/javadoc/org-openide-modules/org/openide/modules/doc-files/api.html/>

[9] NetBeans.org: *Lookup Library - The Solution to Comunication Between Components* [online]. 2009 [citováno 12.února 2009] Dostupné na: <http://openide.netbeans.org/lookup/>

[10] NetBeans.org: *What is a Lookup* [online]. 2009 [citováno 12.února 2009] Dostupné na: <http://wiki.netbeans.org/DevFaqLookup/>

[11] Sun Microsystems, Inc.: *Java SE Desktop Technologies* [online]. 2009 [citováno 15.ledna 2009] Dostupné na: <http://java.sun.com/products/javabeans/>

[12] NetBeans.org: *Lookup library API* [online]. 2009 [citováno 12.února 2009] Dostupné na: <http://bits.netbeans.org/dev/javadoc/org-openide-util/org/openide/util/lookup/doc-files/lookup-api.html/>

[13] NetBeans.org: *Window System API* [online]. 2009 [citováno 12.února 2009] Dostupné na: <http://bits.netbeans.org/dev/javadoc/org-openide-windows/org/openide/windows/doc-files/api.html/>

[14] NetBeans.org: *Nodes API* [online]. 2009 [citováno 12.února 2009] Dostupné na: <http://bits.netbeans.org/dev/javadoc/org-openide-nodes/org/openide/nodes/doc-files/api.html/>

[15] NetBeans.org: *Actions API* [online]. 2009 [citováno 12.února 2009] Dostupné na: <http://bits.netbeans.org/dev/javadoc/org-openide-actions/org/openide/actions/doc-files/api.html/>

A Přílohy uloženy na CD

- zdrojové kódy
- program
- uživatelská příručka
- programátorská příručka